Triangle sinclair users' group

TIMEX

MANAGING EDITOR:
Bill Roberts
EDITOR: Doug Dewey

VOLUME II
NUMBERS 5, 6, 7
MAY, JUNE, JULY 1984

2068 NOTES: A TUTORIAL ON MACHINE CODE PROGRAMMING

by DICK SCOVILLE

Part 3

(Part 4 is included in this issue)

2068 Machine Code Tutorial 3
by Dick Scoville

Here are two little programs (8100-8240 and 8300-8520) that will help you load and peek at machine code. They are completely independent, so if you use both of them with a menu for choosing between them, you could make the sum less, in length, than the two parts. Change these programs as you please: if you dislike the CAPSLOCK business (as I do) then get rid of it and input in lower case, but as you make changes please keep this in mind: y o u will probably be the only person using your program, so don't waste memory and loading time by adding a huge number of prompts and error-trapping routines that y o u don't need.

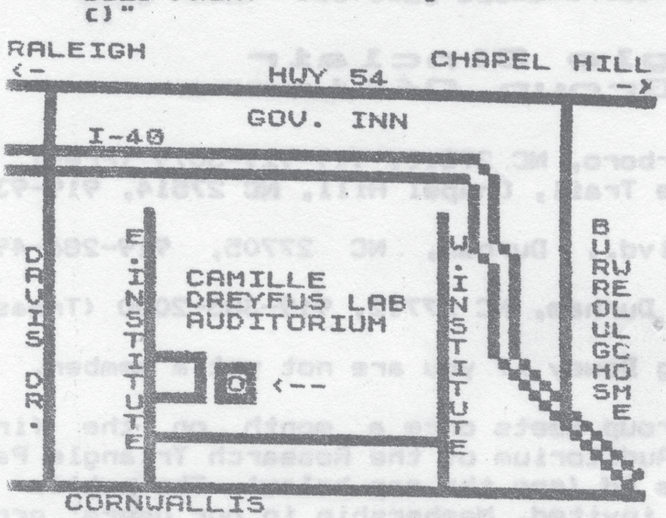
If the first program, rather than printing out the contents, in hex, of successive memory locations, actually decoded the instructions into their mnemonic equivalents and displayed them, it would be a disassembler. If the second program allowed you to type in mnemonics and turned them into code, it would be an assembler. My "mcdis" tape is mainly a disassembler (and many thanks to everybody who ordered it), but if you need a first-rate assembler-disassembler, try Hot Z--it's really nice.

```
8100 REM Machine Code Peeker
8110 LET h$="0123456789ABCDEF"
8120 PRINT "starting address (dec)"
8130 INPUT m
8140 LET n=PEEK m
8150 GO SUB 8200
8160 PRINT m;TAB 7;b$;"    ";n
8170 LET m=m+1: GO TO 8140
8200 LET t=INT (n/16)
8210 LET b$=h$(t+1)
8220 LET t=n-16*t
8230 LET b$=b$+h$(t+1)
8240 RETURN
8300 REM Machine Code Loader
8310 LET h$="0123456789ABCDEF"
8320 PRINT "starting address (dec)"
8330 INPUT m
8340 PRINT "set CAPS LOCK"
8345 PRINT "input hex"
8350 PRINT "append S to quit"
8355 INPUT v$
8360 IF v$="" THEN GO TO 8355
8370 IF v$(1)="S" THEN STOP
8380 LET a$=v$(1)
8390 GO SUB 8500
8400 LET k=16*t
8410 LET a$=v$(2)
8430 GO SUB 8500
8440 LET k=k+t
8450 POKE m,k
8460 LET m=m+1
8470 LET v$=v$(3 TO )
8480 GO TO 8360
8500 LET t=1
8510 IF h$(t)<>a$ THEN LET t=t+1: GO TO 8510
8520 LET t=t-1: RETURN
```

Now back to the tutorial: we've seen how numbers can be written to base 16, and we've known since childhood how numbers can be written to base 10. We need some symbols--sixteen of them in the first case 0123456789ABCDF, and only ten 0123456789 in the second. Carrying the idea to its extreme, consider how numbers might be written with just t w o symbols, say 0 and 1. Here are the first few: 0,1,10,11,100,101,110,111,1000. notice how easy it is to add in this system:

```
   11011101    (=221)
+   1001001    (= 73)
-----------
  100100110    (=294)
```

1 and 1 is 10--write down the 0, carry the 1, etc. What is hex F in this system? 1111, the number just before 10000 (=16 dec). And what is FF? You guessed it: 11111111, 255 decimal, the largest number that can be written with just eight binary digits. Although until now we have thought of each address location or Z80 register as holding "a number between 0 and 255" we see now that in fact, each of these "holes" is really eight tiny "holes"--bits--each of which can contain either a 0 or a 1.



In fact, as I understand it, each of these bits is like a tiny capacitor, either charged (1) or uncharged (0), so we have really come down to the electronic foundations.

But the beauty of the binary system is that 0 and 1 can stand for no and yes, or false and true, or off and on, as well as zero and one. Here are the complete addition and multiplication tables for binary calculations:

+	0	1
0	0	1
1	1	10

X	0	1
0	0	0
1	0	1

and here are three more "operations," besides + and X that can be performed on a pair of binary digits:

OR	0	1
0	0	1
1	1	1

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

The key to understanding how to get the answer in each of these calculations is this: let P stand for the statement, "It's raining," and let Q stand for the statement, "Today's Friday." Notice that you can determine the truth or falsity of the statement, P OR Q, if you know whether P is true or false and whether Q is true or false. For instance, if P is true (1) and Q is false (0) then the statement P OR Q is true, i.e., 1 OR 0 = 1. Similarly, only if P and Q are both true will the statement, P AND Q, be true. XOR (exclusive OR) means one or the other is true, but not both.

(Continued on next page)

Part 5 will appear in August and lets you print sideways on the 2040 printer

Do You Own a 2068? Or Are You Getting One? And Are You Interested In Machine Code Programming?

Undoubtedly you would like to own a nice machine code loader, and a simple disassembler, and a hex/decimal;decimal/hex converter. Then you could easily follow and learn from Dick Scoville's machine language tutorial which starts in this issue. Such utility programs as we have mentioned above usually cost $9.95 and up. BUT for TSUG receivers, Dick will provide all the above programs on a good tape for only $5.00 (s+h included)--a real bargain. Please send your order today to Dick Scoville, 2313 W. Club, Durham, NC 27705. If you purchase this tape,we will consider your purchase to be a show of interest and confidence in TSUG and will not soon drop you from our newsletter mailing list. This is important if you have been getting our newsletter FREE NC.

We have many people who can help you as a member of the users' group. (But please don't ask for a huge amount of help an expect a hasty reply if you aren't participating as a contributing member of this users' group; remember, you participat as you communicate.) The best person to write to for help on the 2068 and on machine code and FORTH and such things a complex routines to do extraordinary things is Dick Scoville. The best person to write to for help of all kinds, help i machine code and help on hardware on the ZX80, ZX81, TS100 is definitely Hi Demby. Hi can also help you with genera hardware questions on the 2068. The best persons to help with sources and general information on particular products ar the editorial staff of Doug Dewey and Bill Roberts. Let me just say that Bill has been doing a fine job of learning th ropes of editing this newsletter. We hope you enjoy what we have come up with. We have a number of reviews next month o products that have been in the hands of our reviewers for quite some time and have been fully tested.

The August meeting will be held on the 5th in Dreyfus Auditorium, as usual, starting at 2:00 pm. The meeting will featur Bill Roberts demonstrating his Delphic chip for the TSZX and a continuation of Jim Hicks and the 64 column mode. All ar welcome.

The September meeting will be on the 2nd, Labor Day weekend, of course. We hope to have lots of new friends come to visi at that time.

Triangle Sinclair Users' Group Officers

Doug Dewey, 206 James St., Carrboro, NC 27510, 919-929-3079 (Pres)
Hiawatha Demby, Jr., 211 Nature Trail, Chapel Hill, NC 27514, 919-933-9362 (Vice)
Dick Scoville, 2313 W. Club Blvd., Durham, NC 27705, 919-286-4959 (Sec)
Mike Boyce, 3313 East Oak Dr., Durham, NC 27712, 919-383-2000 (Treas)

Send mail and inquiries to Doug Dewey if you are not yet a member.

The Triangle Sinclair Users' Group meets once a month on the first Sunday at 2pm in the Dreyfus Auditorium of the Research Triangle Park facility just off of Cornwallis Rd.(see the map below). The public is welcome, in fact the public is invited. Membership in our users' group is $10.00 a year and now is the time to send in your contribution. We have a large public that we serve, and we need sustaining contributions in the form of money, articles for the newsletter, or ocasional letters telling us you are still interested in receiving our publication. The donation of goods and monies by members and business friends alike, is a welcome occurence and is also tax-deductible.

These logical operations have many many uses in machine code--a lot more than they do in BASIC lines. Suppose register A contains a number between 00 and FF let's say D9 (=217). What happens if the Z80 executes the two-byte instruction

```
E6F8     AND F8
code     mnemonic
```

Although the mnemonic doesn't indicate it, logical operations such as this one can be performed only on the contents of the A register. Now

```
A contains D9 which is 11011001
F8                    is 11111000
```

After execution, A is 11011000 An AND operation has been performed bit by bit. The total effect has been to turn the last three bits of A to 0 and leave the top five alone. A similar result would be achieved by

```
06F8  LD B,F8    load B with F8
A0    AND B      AND A with B
```

Remember, the result always ends up in register A.

Just to get the mechanics straight, let's write a program that contains a small machine-code subroutine. First load in something to help you write machine code, for instance the two programs mentioned at the beginning. We want our subroutine to

> take a number betw. 0 and 255,
> divide it by 8,
> give the remainder.

That is, if we input n, we want our routine to give the number n-8*INT(n/8). We have to decide where to put the code we are about to write: let's put it in approximately the middle of the free space of the computer, say at 40000, so CLEAR 39999. Now there is no danger that BASIC will use (overwriting our code) any addresses above 40000. At this point you may want to try out the loader program by putting in some hex values (remember every address should hold a pair of hex digits) and then peeking them to be sure everything is OK. But don't poke below 40000--that's BASIC territory now. Our BASIC lines will look like this:

```
10 PRINT "enter a no., 0 to 255"
20 INPUT n : CLS
30 POKE 40000,n
40 PRINT n,USR 40001
50 GO TO 10
```

In line 40, we don't say USR 40000, because if the Z80 were directed to start reading there it would interpret the number n which we will have entered there as c o d e , and who knows what would happen after that--there would probably be a crash, that is, a situation in which the Z80 busily executes instruction after instruction and we have no way of getting it to return to BASIC.

Now, what about the machine code program starting at 40001? First, we have to "get" the number that lies in address 40000 (=9C40 hex). Here's how:

```
3A409C  LD A,(9C40)
```

This is an instruction type we haven't met before. Take the parentheses to mean "the contents of." This instruction loads register A with the number which is in the hole whose address is 9C40, that is, the contents of address 9C40.

Next, please convince yourself that the number we ultimately want (the remainder) is simply the number made up of the last three binary digits of n--just as, if we were working in decimal and wanted the remainder when 765427 is divided by 1000, the answer would be the last three (decimal) digits, namely, 427. We are dividing by 8 (=1000 binary), so our problem will be solved if we can make the top five binary digits of n 0's and leave the last three alone. That's easy:

```
E607  AND 07
```

This works because 07(hex)=00000111(binary),so AND'ing A with 07 kills (masks) the top five digits of A and leaves the last three unchanged. At this point we have the answer in A--

how do we deliver it to the USR routine in BASIC? Remember that that routine will take as the answer whatever number is in the BC register pair, so we have to put the number in A into BC. That's easy too:

```
4F    LD C,A
0600  LD B,00
```

Now we're ready to hand control back to BASIC, so

```
C9    RET
```

Our routine takes up .1+3+2+1+2+1=10 bytes. Load it in, from 40001 to 40010. (For comparison the BASIC line

```
99 LET r=n-8*INT(n/8)
```

takes up 30 bytes!)

Now peek your code, just to be sure, make sure your BASIC lines are OK and try it out. Mine worked--I hope yours does too.

Unfortunately, these explanations always take more space than I expect, so the s t a c k will have to wait until next month. Also, congratulations if you've managed to get through this article--it was long and hard. Next month's won't be as bad.

Machine Code Tutorial Part 4
Dick Scoville

It's about time we discussed some of the other registers of the Z80 chip. First, there are the registers H, L, D, E, B and C, often used in pairs HL, DE and BC to hold addresses or numbers larger than 255. Also, there is the register A, the accumulator, which has as its mate the register named F, the flags register. Very few instructions refer to F directly, but the contents of F have to be kept in mind almost all the time since the bits of F determine how the program will branch.

Some examples might help. Consider the instruction

```
FE40  CP 40   "hypothetically do A - 40,
              and change
              register F
              accordingly"
```

The CP stands for "Compare," and you should think of the Z80 chip as tentatively considering A - 40 (that is, the contents of A minus the number 40, hex of course). I say tentatively because after the instruction has been executed, A is exactly what it was before--it has not been reduced by 40. But the bits of F have been changed. This is used in instructions such as

```
CAAA31 JP Z,31AA  "jump to
                  31AA if the zero
                  flag is set"
```

Of course the "zero flag" is a certain bit of F and will be set (1) only if A - 40 equals 0, that is, only if A equals 40. Here are some other conditional instructions:

```
C0        RET NZ  "return
              (from the call)
              if not zero"

DC1E03    CALL C,031E  "if
              the carry flag
              has been set,
              call routine at
              031E"

F23F12    JP P,123F  "if
              A-40 is posi-
              tive then jump
              to 123F"
```

The carry flag will be set if, roughly speaking, in the calculation of A - 40, you had to "carry." That will happen only if A is a number less than 40. The P stands for positive or plus. Similar instructions have M, for negative or minus. The numbers from 00 to 7F (0 to 127, decimal) are considered "positive" and the others (from 80 to FF) negative. The positive numbers have binary representations of the form 0xxxxxxx and the negatives have representations of the form 1xxxxxxx. Note that adding FF to a register has the same effect on the register as subtracting 1, for instance

```
   40
+ FF
----
   3F
```

Although F+4=13 (15+4=19, dec), the "carried" 1 of 13 gets dropped. This partially justifies the terminology plus and minus.

Unfortunately, there is no way of deducing what a particular Z80 instruction will do to the flags--you simply have to learn them. Here are some hints that generally hold: LD (load) and JP (jump) don't affect any of the flags; arithmetic-type instructions ADD (add), SUB (subtract) and logical-type instructions OR (or), XOR (exclusive or), AND (and) tend to affect all the flags; instructions involving register pairs tend not to affect any of the flags. Here are some examples:

```
A000 3EF0      LD A,F0
A002 3C        INC A
A003 C200A0    JP NZ,A000

*

A000 1B        DEC DE
A001 7A        LD A,D
A002 B3        OR E
A003 C8        RET Z

*

A000 7E        LD A,(HL)
A001 FE40      CP 40
A003 CA020E    JP Z,0E02
A006 23        INC HL
A007 C300A0    JP A000
```

In each case, try to work out what will happen. If a condition is n o t met, the conditional instruction has no effect and the Z80 goes to the instruction following.

So much for the main line of registers B, C, D, E, H, L, A and F. There are eight more registers B', C', D', E', H', L', A' and F'. They can be thought of as storage bytes. For example,

```
D9    EXX      "get the values
               of B',C',D',E',H',L'
               out of storage and
               put them in B,C,D,E,
               H,L, and put the cur
               rent values of B,C,
               D,E,H,L into stor-
               age"
23    INC HL
D9    EXX
```

The net effect of these three instructions will be to increment the pair H'L' and not change any of the main line registers. Note that the pair AF has an exchange instruction of its own--EX AF,AF'. AF' means A'F.

There are two special purpose register pairs SP and PC--no, SP doesn't stand for special purpose, nor does PC stand for personal computer. SP is the stack pointer and PC is the program counter. The purpose of PC is very clear--to keep track of where the Z80 is supposed to go to execute its next instruction Ordinarily, this will be the instruction following the current one, but what happens in the case of, say,

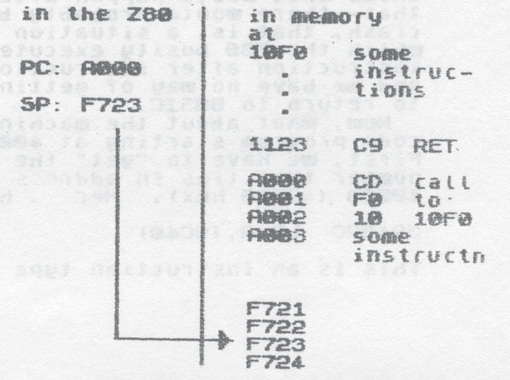
```
A000 C3F010   JP 10F0   ?
```

The Z80 simply loads PC with the address 10F0 and is ready for the "next" instruction. And where is the "next" instruction? As always, at the address contained in PC, so the Z80 successfully makes the jump. But what about this case?

```
A000 CDF010   CALL 10F0
A003 47       LD B,A
```

After execution of the 3-byte CALL instruction at A000, the next instruction executed should be the one at 10F0. But how will the Z80 "remember" that after it jumps to 10F0 and starts executing there, it is supposed to jump back to A003 when it encounters a RET instruction? The answer is: before it jumps, i.e., before it changes PC to 10F0, it "puts the return address on the stack."

Suppose, for example, that before the execution of the CALL at A000, SP contains the address F723. So the situation looks like this:



Now the Z80 begins work on the call instruction at A000: it d e c r e m e n t s SP by 2 and loads F721 and F722 with 03 and A0 (lo-hi of course). It then loads PC with 10F0 and goes to the "next" instruction, that is, the one at 10F0. Assuming it encounters a RET instruction at 1123, what happens? (Remember, at this point SP is F721.) It loads PC with the numbers at SP and SP+1, namely, A000, and i n c r e m e n t s SP by 2. It then goes to the "next" instruction which of course is the one at A003.

So, by the time the Z80 executes the instruction at A003 SP will again be F723. It isn't necessary to "erase" the contents of F721 and F722. They will simply be overwritten the next time an address gets "put on the stack." Notice that the stack grows downward as addresses get put on it (when there are calls within calls, for example) but it is always LIFO: the last address in is the first address out, because SP is always "pointing to" the latest address on the stack.

You yourself can instruct the Z80 to put addresses (or any numbers between 0000 and FFFF) on the stack via instructions of the type

```
E5      PUSH HL
F5      PUSH AF
        etc.
```

In the first case, SP will be decremented by 2 and the memory spots at addresses SP and SP+1 will be loaded with the contents of L and H.

You take something off the stack by "popping" it:

```
D1      POP DE
```

will load DE with the 2-byte number at the bottom of the stack and change SP appropriately. Perversely, some people call what is the bottom of the stack, the "top" of the stack. Anyway, they are always referring to the most recent addition to the stack.

Note that the pair of instructions

```
PUSH HL      (6.29)
POP DE       (5.71)
```

has exactly the same effect as

```
LD D,H       (2.29)
LD E,L       (2.29)
```

but the execution times shown (in millionths of seconds, micro seconds) are rather different. I am assuming that our 2068 clocks run at 1.75 MHz, which I believe is correct.

PHONEBOOK

by Guy Qualls

I have tried a couple of times to get this program to work by holding the receiver up to the computer's speaker and each time it almost automatically dials for me but not quite. I think I need a little amplifier to make the signal stronger. See what you can do with it. It is clever, nonetheless.

```
  1 PRINT "PHONEBOOK Hit any ke
y to start": PAUSE 0
  5 SOUND 7,56;8,0;9,0
  9 CLS : INPUT "Who would you
like me to dial";'b$: RESTORE
 10 CLS : READ x$,a$
 11 IF x$="r" THEN PRINT "no su
ch number": PAUSE 100: GO TO 9
 12 IF x$<>b$ THEN GO TO 10
 15 PRINT x$
 20 FOR f=1 TO LEN a$
 30 PRINT a$(f);
 35 IF a$(f)="p" THEN PAUSE 0
 40 IF a$(f)="1" THEN GO SUB 10
20: GO SUB 1060
 50 IF a$(f)="2" THEN GO SUB 10
20: GO SUB 1070
 60 IF a$(f)="3" THEN GO SUB 10
20: GO SUB 1080
 70 IF a$(f)="4" THEN GO SUB 10
30: GO SUB 1060
 80 IF a$(f)="5" THEN GO SUB 10
30: GO SUB 1070
 90 IF a$(f)="6" THEN GO SUB 10
30: GO SUB 1080
100 IF a$(f)="7" THEN GO SUB 10
40: GO SUB 1060
110 IF a$(f)="8" THEN GO SUB 10
40: GO SUB 1070
120 IF a$(f)="9" THEN GO SUB 10
40: GO SUB 1080
130 IF a$(f)="0" THEN GO SUB 10
50: GO SUB 1070: PAUSE 2
200 PAUSE 3
210 SOUND 8,0;9,0
220 PAUSE 1
998 NEXT f: PRINT : PAUSE 60
999 GO TO 1
1020 SOUND 0,158;1,0;8,15
1025 RETURN
1030 SOUND 0,143;1,0;8,15
1035 RETURN
1040 SOUND 0,130;1,0;8,15
1045 RETURN
1050 SOUND 0,117;1,0;8,15
1055 RETURN
1060 SOUND 2,91;3,0;9,15
1065 RETURN
1070 SOUND 2,83;3,0;9,15
1075 RETURN
1080 SOUND 2,75;3,0;9,15
1085 RETURN
2000 DATA "syntax","16174563661"
2010 DATA "doug","19199293079"
2020 DATA "guy","18062969389"
2030 DATA "anybody","2222222222"
9900 DATA "r","0"
9999 SAVE "PHONEBOOK" LINE 1
```


ZX PRO/FILE
CARTRIDGE
WITH "QUICKLOAD"! $39.95

The most professional file and database manager for the TIMEX—now on plug-in cartridge! Type one command and the program runs instantly. But that's not all—you also get "QUICKLOAD" which saves and loads 16K in just 30 seconds! With 59 page manual! SATISFACTION GUARANTEED.
BOTH on one plug-in cartridge..... $39.95

QUICKLOAD CARTRIDGE $29.95

- Saves/loads 16K in 30 seconds
- Stores over a megabyte on one cassette
- Next best thing to a disk drive

- Check or money order
- Shipping add $ 4
- Cal. res. add 6½ %
- SASE for catalogue

Both cartridges have rear expansion port for 16/64K RAM.

For the LARGEST selection of EPROM software

Rompak
TS1000 1500
8206 Blackburn Ave.
Los Angeles, CA 90048

UNDERSTANDING SINCLAIR BASIC

BY Alan Porter

Alan Porter was kind enough to release his program through us to reside in the public domain. We know it does not contain the machine code material in this issue. If you would like a copy of the whole program, recorded twice on a fine quality tape, send us £5.00 and we will mail it back to you. This program is for the TS2X computer with 16K of memory. It is elementary in its approach but a very fine effort from an outstanding programmer. It is a nice review of good practices in programming and could serve as a very useful tool in a program of computer education centered around the TS2X line of machines--most especially for its use as a demonstration of good style.

WHAT IS UNDERSTANDING SINCLAIR BASIC?

Understanding Sinclair Basic is a 16k tutorial program for demonstrating the TS/1000 Basic language. It is written in machine language and Basic. U.S.B. is a menu driven program. It allows a user to see the Basic commands, statements,and syntax within the contextual framework of a running program. It does this in a clear "user friendly" manner while locking out or minimizing errors. U.S.B. is not a manual, rather it is designed to function with a manual as a learning aid. It directly addresses the standard problem in manual presentation of confusing English grammar with computer syntax.

WHAT DOES IT DO?

The heart of U.S.B.'s presentation and it's overriding strength is it's ability to display a line of Basic, pause for a user cue, then run the same line of basic under the listing. This split screen effect allows the user to see what each line does in a program listing. The selected program can be repeated immediately for continued study or the user may return to the main menu for another command. The user thus experiences immediate feedback in his own time framework as opposed to a time based on the speed of the Basic interpreter or Z80 processor.

HOW DOES IT WORK?

Upon entering U.S.B. the user is presented with two menu screens which display the Basic language which U.S.B. supports. The user selects the statement which he wishes to see demonstrated and is shown a listing of the demonstration program. The program waits at this point for the user to press Enter. The user can thereby study the program listing or make notes. A copy of the listing can be obtained by pressing the "C" key. After pressing Enter or "C", U.S.B. displays each line consecutively while waiting for a user prompt. The program lines are listed and individually run on the same screen. The user controls the speed at which they are run because the program pauses after each line is run. When the last line is run U.S.B. prompts to repeat the program or for a return to the menu.

INSTRUCTIONS FOR UNDERSTANDING SINCLAIR BASIC

LOADING: (Load time: 7 minutes)

Load U.S.B. by pressing the J or load key (K cursor must be showing) then, holding down shift, type two quotes.

EXAMPLE: LOAD "" optional load... (LOAD "USB")

ADJUSTING VOLUME LEVEL:

It is very important with this program that you establish the proper loading level because it takes about 7 minutes to load.

A good method of adjusting your volume level is to start with the volume on the maximum setting then initiate a load. After the program is loading then move the volume slowly lower until the loading pattern drops out or disappears. Note this point and move the volume half way between that setting and maximum. Restart the load at your new setting. A good rule of thumb is "louder is generally better" but too loud sometimes distorts. If you get a loading pattern but the program doesn't start after the load then try another load at a slightly higher volume.

Once you get a good loading level, make a note of it for future reference. U.S.B. has been saved once on each side of the cassette.

ABOUT THE PROGRAM

Understanding Sinclair Basic is a menu driven program which means that you choose a word from a menu screen and U.S.B. takes you to the program. After loading U.S.B. you will be welcomed by a title page. Press Enter and you will be in the menu. At this point you have two choices (which are listed at the bottom of the screen). All other keys are locked out or dead.

Press "M" and you will see the second page of the menu section. Press "M" again and you will be returned to page one. Try holding down "M" to see what happens.

Press "L" and you will be prompted to enter a number. Choose a word (try CLS) and enter the number. Press Enter. If you enter only one number (ie. 5 instead of 05) the program will reject your choice. It will not accept a letter or more than two digits either.

After you have entered the correct number and pressed Enter, the screen will go blank briefly and you will see the demonstration program on the screen. This is to give you an overview of the program and to allow you to make notes. Press "C" to make a copy of the listing or press Enter to run the program. After pressing "C" or Enter you will be shown the first line of the program. Press Enter again to run that line and the second line will appear. Press Enter again, etc. At the finish of the program an instruction will appear at the bottom of the screen giving you two choices.

Press "M" to return to the menu for another selection.

Press "R" to repeat the program.

Occasionally, you will be asked for an input. A cursor will appear or a pair of quotation marks. Enter a number or a word depending on the type of prompt. (see the manual on INPUT).

Most errors have been locked out of U.S.B. but if you should make an error (ie. entering a letter into a numeric input) and press Enter, then, because of the operating system you will see your screen filll with the first line of the monitor program which is a large REM statement filled with machine code. If this should happen then enter GOTO 1. DO NOT enter RUN 1 because this will delete all the variables. The program should start back up beginning with the introduction. In general when in doubt just press Enter. Or, if you inadvertently jump out of U.S.B. then enter GOTO 1.

You will also occasionally come across a statement like: 530 GOSUB DEMO. This is a subroutine which is accessed directly by U.S.B. This line is only a note to let you know what is happening. GOSUB like any other Sinclair command can be used with a proper variable name. For a complete explanation on GOSUB, see your manual or see the GOSUB routine in U.S.B.

By carefully examining the Basic lines and watching what they do you will quickly begin to understand Basic on your TS/1000. Understanding Sinclair Basic is NOT a substitute for a good manual. Rather, it should be used with a manual.

UNDERSTANDING SINCLAIR BASIC (BASIC SUBROUTINES)

LINE		EXPLANATION
6	:	Poke 16418,2 to prpare bottom two lines
8	:	Goto 23 (skip over menu variable check)
9-22	:	Check for variables and merge routines
23	:	Fill screen (user address is a jump to 16678) (part of lock)
24	:	Goto title printing subroutine; lines 106-111
26	:	Generate a flashing boarder
28	:	Check for variable set in Copy routine: if not COPY then goto 30
30	:	Rapid Screen Clear
35	:	Transfer screen #1
40	:	Enter menu control user routine : Inkey$ = "M" or "L"
50	:	Input of Program number
55-66	:	Check string for proper parameters
70-71	:	Check variable and merge routines
78	:	Gosub lines 9-22 (check variables and merge routines)
79	:	Set variable A as pointer to Program area
80	:	Poke into machine routine the start of first line to be listed
81	:	Pick the number of lines in program that will be listed
82	:	List out the program to be demonstrated FOR/NEXT LOOP
83	:	Print at 4,0; (starting line for Basic)
84	:	Poke starting line again (line 80) because start line is corrupted
85	:	Start 'RUN a BASIC LINE ROUTINE' : pause for user prompt
87	:	Goto lst line of demonstration program
90-99	:	Messages used in PAUSE demonstration
101-105	:	Repeat Routine or Return to Menu for another selection
102	:	Inkey$ = "R" or "M"
105	:	Goto 80 (Reset all variables to relist and rerun program).
106-111	:	Print opening screen titles (line 24)
120-121	:	Print 'Press Enter' , used in User routine demonstrations
130-136	:	Poke machine program with starting line (lines 80 & 84)
140-154	:	List out demonstration program and pause (line 82)
140	:	Rapid Clear Screen
141	:	Position the first line on the screen (depends on lenght of demo)
142-145	:	For/Next loop (lists out program)
143	:	List one Basic line
144	:	Add 5 to line number before listing next line
150	:	Long timer
151	:	Inkey$ (standard anykey)
153	:	Rapid Clear Screen
155	:	Position Printing line used in GOTO demo and after a CLS
166-167	:	Run a BASIC line (LET H=USR 16800)
170-185	:	Messages used in PRINT demonstration
188-189	:	Rapid Screen Clear (lines 30, 140, 153)
190-195	:	Position lst Print line (line 141)
196-197	:	Message used in looping demonstrations
200-9550	:	Body of demonstration programs
9999	:	Rem statement, poked invisible, containing jump to Screen Fill

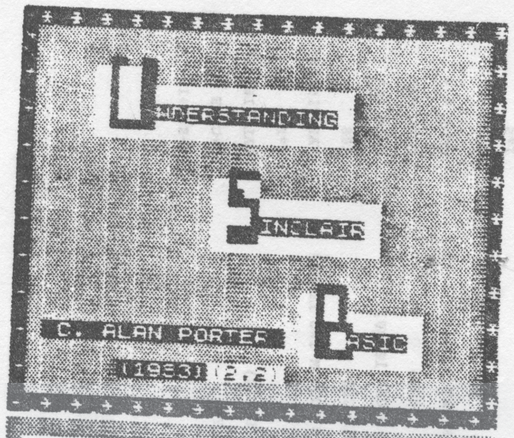
TO REARM LOCK AND MAKE A COPY..... POKE 18675,62 then GOTO 9000

UNDERSTANDING SINCLAIR BASIC (MACHINE ROUTINES)

ROUTINE	DECIMAL ADDRESS	LENGHT HEX	POKE ADDRESS
GRAPHIC BOARDER	16514	4082-410D	
CLS	16654	410E-4125	
FILL SCREEN	16678	4126-413D	
ONE BASIC LINE	16702	413E-415F	413F, 4140 (16703 L, 16704,H)
ADD +5	16736	4160-416A	
TRANSFER SCRN #1	16767	417F-418C	
TRANSFER SCRN #2	16892	41FC-4209	
INKEY$	16787	4193-419F	
RUN LISTING	16800	41A0-41FB	41D7 COMPARE BYTE (MINUS LINE VALUE)
SHORT TIMER	16747	416B-4174	
LONG TIMER	16757	4175-417E	
STORAGE DFCC	16781	418D-418E	
STORAGE SPOSN	16783	418F-4190	
STORAGE CORDS	16785	4191-4192	
SCREEN #1	16906	420A-4500	
SCREEN #2	17665	4501-47F7	
INKEY$ C,L SCR#2	18424	47F8-483C	
INKEY$ C,L SCR#1	18451	4813-483C	
INKEY$ M,R	18495	483F-4855	
GRAPHIC SUBROUTINE	18588	489C-48CB	
CLEAR ONE LINE	18581	4895-489B	
PRINT ENTER LINE	18546	4872-4874	DATA 4875-4894
SAVE ROUTINE	18643	48D3-48ED	
PROGRAM LOCK	18675	48F3-48FE	POKE 18675,62 (TO REARM)

UNDERSTANDING SINCLAIR BASIC (VARIABLE LIST)

VARIABLE DEFINITIONS		SINCLAIR DEFINITIONS
Q=0		SGN 0
O=1		SGN 1
W=2		O+O
P=3		INT PI
I=5		P+W
R = 7		I+W
K = 9		R+W
S = 10		I+I
T = 20		S+S
Y = 50		I*S
U = 100		I*T
L = 155	(PRINT AT 3,0)	CODE "."
X = 160	(USR 16800)	CODE "4"
Z = 166	(LET N=USR 16800)	CODE "A"
G = 188	(RAPID CLS)	188
CL= 16418	(POKE BOTTOM TWO LINES)	VAL "16418"
V = 16600		U*Z
J = 16654	(RAPID CLS)	VAL "16654"
E = 16787	(INKEY$)	J+CODE "1"
F = 16692		E-K*S+I




```
(01) AND                (18) INPUT
(02) AT                 (19) LEN
(03) BREAK              (20) LET
(04) CHR$               (21) LIST
(05) CLS                (22) LLIST
(06) CLEAR              (23) LOAD
(07) CODE               (24) LPRINT
(08) CONT               (25) NEW
(09) COPY               (26) NOT
(10) DIM                (27) OR
(11) FAST/SLOW          (28) PAUSE
(12) FOR/NEXT           (29) PEEK
(13) GOSUB/RETURN       (30) PLOT
(14) GOTO               (31) POKE
(15) GRAPHICS           (32) PRINT
(16) IF/THEN            (33) RAND
(17) INKEY$             (34) REM
```

(INPUT NUMBER)

```
(35) RND        (51) MATH
(36) RUN
(37) SAVE
(38) SCROLL
(39) STEP
(40) STOP
(41) STR$
(42) TAB
(43) TO
(44) UNPLOT
(45) USR
(46) VAL
(47) "
(48) ""
(49) ;
(50) ,
```

NOTE: TRIGONOMETRIC FUNCTIONS ARE NOT INCLUDED IN THE MATH DEMONSTRATION.

UNDERSTANDING SINCLAIR BASIC C.1983

(INPUT NUMBER)

```
  6 POKE CL,0
  8 GOTO T+8
  9 IF A$>="47" AND A$<="50" OR
A$="18" THEN LET A$="32"
 10 IF A$="42" THEN LET A$="02"
 12 IF A$="27" OR A$="16" OR A$
="26" THEN LET A$="01"
 13 IF A$="31" THEN LET A$="29"
 14 IF A$="45" THEN LET A$="34"
 15 IF A$="36" THEN LET A$="40"
 16 IF A$="22" THEN LET A$="21"
 17 IF A$="44" THEN LET A$="30"
 18 IF A$="19" OR A$="43" OR A$
="46" THEN LET A$="41"
 19 IF A$="39" THEN LET A$="12"
 20 IF A$="35" THEN LET A$="33"
 22 RETURN
 23 RAND USR VAL "31574"
 24 GOSUB CODE "?"
 25 PRINT AT T+P,0 "      PRESS
ENTER TO BEGIN      "
 26 RAND USR VAL "16514"
 27 POKE CL,0
 28 IF U=W*Y THEN GOTO T+5
 29 RETURN
 30 RAND USR J
 35 RAND USR 16767
 37 PRINT AT T+P,0;"  (L) LIST
OR     (M) MENU      "
 40 RAND USR 16424
 45 POKE CL,W
 47 PRINT AT T+0,R;"  (INPUT NUM
BER)  "
```

```
 50 INPUT A$
 55 IF LEN A$>W OR LEN A$<W THE
N GOTO Y
 60 IF CODE A$<I*I+P OR CODE A$
>I+R+P THEN GOTO Y
 65 IF CODE A$(W)<I*I+P OR CODE
A$(W)>I*R+P THEN GOTO Y
 66 IF VAL A$>Y+O THEN GOTO Y
 68 FAST
 70 IF A$="14" THEN LET A$="8"
 71 IF A$="20" THEN LET A$="06"
 76 GOSUB K
 79 LET A=VAL A$+U+U+5
 80 GOSUB U+T+S
 81 GOSUB A-S
 82 GOSUB P*Y-S
 83 GOSUB P*Y+I
 84 GOSUB U+T+S
 85 GOSUB X
 86 GOTO A
 90 PRINT AT T+0,0;"  TIMING  "
 91 RETURN
 93 PRINT AT T+0,0;"  DONE  "
 94 PRINT AT W*P,0;
 95 RETURN
 97 PRINT AT T+0,0;"          "
 98 PRINT AT W*P,0;
 99 RETURN
101 PRINT AT T+0,0;"  PRESS (M)
MENU  OR  (R) REPEAT  "
102 RAND USR 18495
103 IF INKEY$="M" THEN GOTO T+R
104 FAST
105 GOTO CODE "?"
106 PRINT AT W,I+0;"  ";AT P,I
;"  ";AT W+W,I;"  ";AT P,I
;"  ";TAB I;"  UNDERSTANDING",TAB
I;"  "
107 PRINT AT R+0,W*R;"  ";AT K,
S+P;"  ";AT S,S+P;"  ";AT K,
TAB S+P;"  INCLAIR",TAB S+P;"  "
108 PRINT AT W*R,T;"  ";AT S+I,
T-O;"  ";AT R+K,T-O;"  ";T
AB W;"  C. ALAN PORTER  BASIC";
TAB T-O;"  "
109 PRINT AT T-O,R;"  (1983) (2,2)
"
110 RETURN
111 RAND USR VAL "16654"
115 RETURN
120 PRINT AT T-I,K;"(PRESS ENTE
R)"
121 RETURN
130 LET H=INT (A/256)
131 LET L=A-256*H
132 POKE F+O,L
133 POKE F+W,H
135 LET L=CODE "  "
136 RETURN
140 GOSUB G
141 GOSUB CODE "  "
142 FOR N=1 TO B
143 RAND USR VAL "16702"
144 RAND USR 16736
145 NEXT N
146 PRINT AT T+0,0;"  PRESS ENT
ER TO START PROGRAM  "
147 SLOW
149 RAND USR E-S-T
150 RAND USR E
151 IF INKEY$="O" THEN COPY
152 GOSUB G
153 RETURN
155 PRINT AT W+W,0;
157 RETURN
166 LET H=USR VAL "16800"
167 RETURN
170 PRINT AT T+0,0;"  INPUT NUMB
ER  "
175 RETURN
176 PRINT AT T+0,0;"
  "
177 RETURN
180 PRINT AT T+0,0;"  INPUT NAME
  "
185 RETURN
186 RAND USR J
189 RETURN
190 PRINT AT W,0;
191 IF A$="33" THEN PRINT AT 0,
0;
195 RETURN
196 PRINT AT P,P;"  PPESS AND
HOLD  "
197 PRINT
199 RETURN
200 LET B=S+O
202 RETURN
210 REM  AND /NOT /OR :ALL CAN
USED IN IF/THEN STATEMENTS
```

```
212 GOSUB X
215 LET N=10
217 GOSUB X
220 LET I=5
222 GOSUB X
223 PRINT
225 IF N=10 AND I=5 THEN PRINT
"TRUE"
227 GOSUB X
230 IF N=12 AND I=5 THEN PRINT
"TRUE"
232 GOSUB X
235 IF N=12 OR I=5 THEN PRINT "
TRUE"
237 GOSUB X
240 IF N=12 AND NOT I=7 THEN PR
INT "TRUE"
242 GOSUB X
245 IF NOT N=12 AND NOT I=7 THE
N PRINT "TRUE"
247 GOSUB X
250 LET B$="CATS"
252 GOSUB X
255 LET A$="DOGS"
257 GOSUB X
260 IF B$<>A$ AND B$<A$ THEN PR
INT "ANIMALS"
265 GOTO U
300 LET B=K
302 RETURN
310 REM  AT/ TAB  USED TO PRINT
TO SPECIFIC SCREEN POINTS
312 GOSUB X
315 LET A$="HERE"
317 GOSUB X
320 PRINT AT 20,20;A$
321 GOSUB L
322 GOSUB X
325 PRINT AT 7,4;A$
326 GOSUB L
327 GOSUB X
330 PRINT AT 5,16;A$;AT 4,0;A$;
AT 19,0;A$;AT 13,24;A$
331 GOSUB L
332 GOSUB X
335 CLS
336 GOSUB L
337 GOSUB X
340 PRINT TAB 0;A$
341 GOSUB L
342 GOSUB X
345 PRINT TAB 7;A$
346 GOSUB L
347 GOSUB X
350 PRINT TAB 20;A$
355 GOTO U
400 LET B=W+W
402 RETURN
410 REM  BREAK  USED TO ESCAPE
A BASIC PROGRAM
412 GOSUB X
415 REM  OR WHEN YOU WISH TO
STOP AT A PARTICULAR POINT
417 GOSUB X
420 REM  ERROR "D" WILL
RESULT
422 GOSUB X
424 GOTO VAL "431"
425 GOSUB  INFINITE LOOP
431 PRINT AT T,0;"NOTE- YOU MUS
T PRESS BREAK AND      THEN EN
TER ""GOTO 435"""
432 PRINT AT S,S;"  LOOPING  "
433 PRINT AT S,S;"
434 GOTO VAL "432"
442 GOTO U
500 LET B=P+W
502 RETURN
510 REM  CHR$  GIVES CHARACTER
STRING OF GIVEN NUMBER
512 GOSUB X
515 PRINT CHR$ 151,CHR$ 138
517 GOSUB X
520 IF CHR$ 8="  " THEN PRINT
"TRUE"
523 GOSUB X
525 CLS
527 GOSUB X
529 GOTO VAL "531"
530 GOSUB  DEMO
  PRINTS CHARACTER SET
531 GOSUB CODE "CODE "
533 LET B=0
536 PRINT CHR$ B;
537 LET B=B+O
538 RAND USR E
539 IF B=K*R THEN LET B=CODE "
"
550 IF B=U+L+O THEN GOTO U
560 GOTO I*U+I*R
600 LET B=I
602 RETURN
```

```
610 REM  CLS: CLEARS THE SCREEN
WITHOUT CHANGING VARIABLES
612 GOSUB X
615 LET A$="  WORRIES  "
617 GOSUB X
619 GOTO VAL "621"
620 GOSUB THINKING
621 PRINT A$;AT P+W,K+R;A$;AT K
+R,T+W;A$;AT K+R,S;A$;AT R+O,T+P
;A$;AT S+W,P;A$;AT K,W;A$;AT W*K
,S+R;A$
624 GOSUB X
625 CLS
626 GOSUB L
629 GOSUB X
630 PRINT AT 10,0;"A$ STILL EQU
ALS ";A$;AT 19,0;"(VARI. ARE NOT
 AFFECTED BY CLS)"
642 GOTO U
700 LET B=K-O
702 RETURN
710 REM  CLEAR  CLEARS ALL
VARIABLES  LET DEFINES VARIBLES
712 GOSUB X
715 LET B$="DOGS"
717 GOSUB X
720 LET M=75
722 GOSUB X
725 PRINT B$
727 GOSUB X
730 PRINT M
732 GOSUB X
733 LET B$="740"
734 GOTO K**P+P+I
735 CLEAR
737 GOSUB X
738 GOSUB R*U+Y+I
739 GOTO R*U+Y-K
740 PRINT B$
741 GOSUB L
742 GOSUB X
743 GOSUB R*U+Y+I
744 GOTO VAL "746"
745 PRINT M
752 GOTO U
755 POKE CL,0
760 PRINT AT T+P,0;"2/";E$
763 PRINT AT K+K+(B$="745"),0;"
(ERROR 2 - UNDEFINED VARIABLE!"
765 LET B$="745"
766 POKE CL,W
767 RETURN
770 GOSUB X
800 LET B=R
802 RETURN
810 REM  CODE  IS THE CODE NO.
OF THE FIRST BYTE IN A STRING
812 GOSUB X
815 LET A$="TREES"
817 GOSUB X
820 PRINT A$,CODE A$
822 GOSUB X
825 LET A$="BUSHES"
827 GOSUB X
830 PRINT A$,CODE A$
832 GOSUB X
835 REM CODE IS ALWAYS A NUMBER
BETWEEN 0-255, IT CAN BE USED
IN EXPRESSIONS
837 GOSUB X
840 IF CODE A$=39 THEN PRINT
"PLANTS"
852 GOTO U
900 LET B=K-O
902 RETURN
910 REM  CONT  WILL ALOW YOU TO
CONTINUE ON ERROR "B"
912 GOSUB X
915 REM  ERROR "B" OCCURS WHEN
SCREEN IS FULL AND PROGRAM STOPS
917 GOSUB X
918 LET END=Y*T-Y
920 LET A=1
922 GOSUB X
923 IF A=T-W THEN PRINT AT T+0,
0;"*FULL* ENTER ""CONT"" TO CONT
INUE"
925 PRINT "REPEAT A=";A
926 IF A=T-W THEN PRINT AT W+W,
0;"REPEAT A=";A
927 GOSUB X
930 LET A=A+1
932 GOSUB X
935 IF A=19 THEN GOTO END
937 GOSUB X
940 REM  GOTO USED FOR BRANCHING
941 POKE E+R*K+I,T+I
942 GOSUB X
943 GOTO VAL "922"
945 GOTO 925
956 LET A=K*U+S
960 GOTO U
```

```
1000 LET B=W+W
1002 RETURN
1010 REM  COPY  PRINTS A COPY OF
 THE SCREEN ON THE PRINTER
1012 GOSUB X
1015 LET U=I
1014 GOTO VAL "1017"
1015 GOSUB OPENING
     PRESS ENTER TO COPY
1017 GOSUB I
1020 COPY
1021 GOSUB L
1022 POKE F+O,O
1023 POKE F+W,W+W
1024 GOSUB X
1025 CLS
1042 LET U=W*Y
1043 GOTO U
1047 GOSUB X
1100 LET B=P+I
1102 RETURN
1110 REM  DIM:  WILL DIMENSION
 A VARIBLE E INTO AN ARRAY
1112 GOSUB X
1113 GOTO VAL "1117"
1115 DIM X$(3,5)
1117 GOSUB X
1116 GOSUB X+T
1119 GOSUB L
1120 INPUT X$(1)
1122 GOSUB X
1125 INPUT X$(2)
1127 GOSUB X
1130 INPUT X$(3)
1131 GOSUB CODE "A"
1132 GOSUB L
1133 GOSUB X
1135 PRINT X$(3)
1137 GOSUB X
1140 PRINT X$(2)
1142 GOSUB X
1145 PRINT X$(1)
1148 GOTO U
1200 LET B=R-O
1202 RETURN
1210 REM  FAST / SLOW  TWO MODES
 WHICH CONTROL DISPLAY + SPEED
1212 GOSUB X
1215 SLOW
1217 GOSUB X
1219 GOTO VAL "1221"
1220 GOSUB PRINT
1221 GOSUB VAL "9500"
1222 GOSUB X
1225 CLS
1226 GOSUB L
1227 SLOW
1228 GOSUB X
1230 FAST
1231 SLOW
1232 GOSUB X
1233 FAST
1234 GOTO VAL "1236"
1235 GOSUB PRINT
1236 GOSUB VAL "9500"
1237 SLOW
1238 GOTO U
1290 PRINT
1300 LET B=R
1302 RETURN
1310 REM  FOR/NEXT  A LOOPING
 STATEMENT  STEP  CONTPOLL VALUE
1312 GOSUB X
1315 LET A$="COMPUTER"
1317 GOSUB X
1318 GOSUB CODE "B"
1319 PRINT AT T-I,W;"(ENTER STEP
 VAULE 1 TO 10)"
1320 INPUT ST
1321 CLS
1322 GOSUB L
1323 IF ST>5 OR ST<O THEN GOTO V
AL "1320"
1324 GOSUB X
1325 PRINT "STEP =";ST
1328 GOSUB X
1330 FOR N=1 TO 10 STEP ST
1332 GOSUB X
1335 PRINT A$
1337 POKE E+R*K+I,5
1338 GOSUB X
1339 NEXT N
1355 GOTO U
1400 LET B=P+I
1402 RETURN
1410 REM  GOSUB  USED FOR
 REPETITIVE SUBROUTINES
1412 GOSUB X
1415 LET B$="PROGRAM"
1417 GOSUB X
1420 PRINT B$
1422 GOSUB X
1425 PRINT B$
1426 GOSUB X
1427 LET B=W
1428 LET A=VAL "1470"
1429 GOTO A-I
1430 GOSUB 1470
1432 GOSUB X
1435 PRINT B$
1436 GOSUB X
1437 LET B=0
1438 LET A=VAL "1470"
1439 GOTO A-I
1440 GOSUB 1470
1442 GOSUB X
1445 PRINT "END OF ";B$
1452 LET A=VAL "1410"
1455 GOTO U
1465 GOSUB U+T+5
1468 GOSUB X
1470 PRINT "SUBROUTINE
1472 GOSUB X
1475 PRINT "*"
1476 GOSUB X
1477 LET A=VAL "1435"+(B=0)*5
1478 GOSUB U+T+5
1479 GOTO A-P
1480 RETURN
1600 LET B=P
1602 RETURN
1610 REM  THIS IS A GRAPHICS
 DEMONSTRATION,,
1612 GOSUB X
1615 REM  IN MACHINE LANGUAGE
1617 GOSUB X
1620 RAND USR 16586
1622 GOTO U
1800 LET B=R
1802 RETURN
1810 REM  INKEY$  RETURNS THE
 VALUE OF KEY BEING PRESSED
1812 GOSUB X
1815 FOR N=1 TO 5
1816 GOSUB X
1817 PRINT AT R+O,S+P;"      ";AT
 S+W,S+P;"      ",,,,"(EXCEPT BRE
AK)"
1818 PRINT AT W+O,O;"( PRESS ANY
 KEY TO PRINT LINE )"
1819 RAND USR VAL "16787"
1820 PRINT AT 10,15;INKEY$
1821 POKE E+R*K+I,S
1822 GOSUB X
1825 NEXT N
1826 LET A=VAL "1830"
1827 GOSUB U+T+5
1829 GOSUB X
1830 CLS
1831 GOSUB L
1832 GOSUB X
1835 REM  INKEY$ IS OFTEN USED
 IN IF/THEN STATEMENTS
1836 GOSUB X
1837 PRINT AT W+O,O;"   ( PRESS "
"C"" TO PRINT LINE )    "
1838 RAND USR VAL "16787"
1839 PRINT
1840 IF INKEY$="C" THEN PRINT "
INKEY$ = "C"
1841 LET A=VAL "1810"
1842 GOTO U
2200 LET B=I
2202 RETURN
2210 REM  LIST / LLIST  BOTH USED
 FOR LISTING PROGRAM
2212 GOSUB X
2213 PRINT
2215 LIST 9500
2217 GOSUB X
2220 CLS
2222 GOSUB X
2225 REM  LLIST WILL LIST OUT ON
 THE PRINTER
2227 GOSUB X
2240 LLIST 9500
2250 GOTO U
2400 LET B=P
2401 RETURN
2410 REM  LOAD  USED TO TRANSFER
 PROGRAMS FROM TAPE
2412 GOSUB X
2415 REM  LOAD IS WRITTEN
 LOAD ""   OR   LOAD "ANY WORD"
2416 PRINT AT S+W,O;"THIS DEMO W
ILL NOT STOP BY","ITSELF, YOU MU
ST PRESS BREAK"," THEN GOTO 2425
",TAB I;"          "
2418 GOSUB X
2420 LOAD ""
2425 GOTO U
2500 LET B=W
2502 RETURN
2510 REM  LLPRINT  THE SAME AS
 PRINT BUT PRINTS ON THE PRINTER
2512 GOSUB X
2515 LPRINT "  THIS IS A DEMO OF
 LLPRINT "
2517 GOTO U
2600 LET B=P+O
2602 RETURN
2610 REM  NEW  WILL ERASE YOUR
 ENTIRE PROGRAM
2612 GOSUB X
2615 REM  ANYTHING PUT ABOVE
 RAMTOP WILL BE UNAFFECTED
2617 GOSUB X
2620 REM  YOUR BASIC PROGRAM WILL
 BE LOST FOREVER
2622 GOSUB X
2623 GOTO VAL "2626"
2625 NEW
2626 CLS
2627 FOR N=O TO T
2628 NEXT N
2629 PRINT AT R+K,P;"YOU JUST ER
RASED",TAB P;"THE ENTIRE PROGRAM
..."
2630 FOR N=K TO T
2631 NEXT N
2632 PRINT AT T+O,P;"  PRESS ENTE
R TO RECOVER "
2634 RAND USR E
2635 GOSUB G
2636 GOTO O
2900 LET B=S
2901 RETURN
2910 REM  PAUSE  USED FOR TIMER
 AND PROGRAM INTERUPT
2912 GOSUB X
2915 PRINT "ENTER A TIMING INTER
VAL","(1 TO 40000):";
2917 GOSUB X
2920 INPUT N
2922 GOSUB X
2925 PRINT N
2927 GOSUB X
2929 GOSUB K*S
2930 PAUSE N
2931 GOSUB U-R
2933 GOSUB X
2935 REM  PAUSE CAN BE USED IN
 CONDITIONAL STATEMENTS
2936 GOSUB U-P
2938 GOSUB X
2940 LET M=1
2942 GOSUB X
2944 GOSUB K*S
2945 IF M=1 THEN PAUSE N
2946 GOSUB U-R
2949 GOSUB X
2950 REM  PAUSE 40000 IS FOREVER
 OR UNTIL A KEY IS PRESSED
2951 GOSUB U-P
2953 GOSUB X
2954 PRINT AT T+O,O;"(PRESS ENTE
R TO BREAK PAUSE)"
2955 PAUSE 40000
2957 GOTO U
3000 LET B=S
 001 RETURN
3010 REM  PEEK  DISPLAY BYTE AT
 ADDRESS  POKE  ENTER BYTE TO
 ADDRESS
3012 GOSUB X
3015 PRINT PEEK 16653
3017 GOSUB X
3020 POKE 16653,61
3022 GOSUB X
3025 PRINT PEEK 16653
3027 GOSUB X
3030 REM  BOTH CAN BE USED IN
 EXPRESSIONS,,
3032 GOSUB X
3035 PRINT PEEK 16385+256*PEEK 1
6389
3037 GOSUB X
3038 RAND
3040 IF A$="29" THEN POKE 16653,
20
3042 GOSUB Z
3045 REM PEEKED NUMBER IS ALWAYS
 BETWEEN 0-255 ... NUMBER POKED
 IN IS ALSO BETWEEN 0-255
3047 GOSUB Z
3050 REM PEEK A RANDOM ADDRESS,,
3051 GOSUB Z
3055 PRINT PEEK INT (RND*32000)+
1
3062 GOTO U
3100 LET B=5+O
3102 RETURN
3110 REM  PLOT/UNPLOT  USED IN
 GRAPHS AND PLOTING TO THE SCREEN
 112 GOSUB X
3115 LET M=0
3117 GOSUB X
3120 FOR N=10 TO 15
3122 GOSUB X
3125 PLOT N,M
3127 GOSUB X
3130 LET M=M+2
3131 IF M<VAL "22" THEN POKE E+R
*K+I,S+I
3134 GOSUB X
3135 NEXT N
3137 GOSUB X
3140 LET M=0
3142 GOSUB X
3145 FOR N=10 TO 15
3147 GOSUB X
3150 UNPLOT N,M
3152 GOSUB X
3155 LET M=M+2
3156 IF M<22 THEN POKE E+R*K+I,5
+I
3157 GOSUB X
3160 NEXT N
3165 GOTO U
3170 STOP
3300 LET B=S+P
3302 RETURN
3310 REM  PRINT  ;  ;" ";" "  ALL
 ARE USED TO PRINT TO THE SCREEN
3312 GOSUB X
3315 PRINT "TS/1000"
3317 GOSUB X
3320 PRINT "TS/1000","""TS/1000"
""
 322 GOSUB X
3325 PRINT 12+12
3327 GOSUB X
3330 PRINT "12+12",12+12
3331 GOSUB X
3335 REM  PRINT WILL ALSO PRINT
 VARIABLES (WORDS + NUMBERS)
3337 GOSUB X
3340 REM INPUT  USED TO ENTER
 NAMES OF NUMBERS TO A PROGRAM
3342 GOSUB X
3343 GOSUB CODE "B"
3344 PRINT AT R,O;
3345 INPUT A$
3347 GOSUB X
3348 GOSUB CODE "B"
3350 INPUT N
3351 GOSUB CODE "A"
3352 PRINT AT R+O,O;
3353 GOSUB X
3355 PRINT A$,N
3357 GOSUB X
3360 PRINT A$;A$,N;N
3362 GOSUB X
3365 PRINT N;"*";N;"=";N*N
3367 GOSUB X
3370 PRINT "DO YOU UNDERSTAND PR
INT NOW?",A$
3387 GOTO U
3400 LET B=R+I
3403 RETURN
3410 REM  RAND  A SEED FOR RND
 AND A USEFUL MACHINE CODE START
3412 GOSUB Z
3415 REM  RAND AS A SEED
3417 GOSUB Z
3420 LET N=525
3422 GOSUB Z
3425 REM WITH RAND SET TO NUMBER
 RND WILL  NOT RANDOMIZE
3427 GOSUB Z
3430 RAND N
3432 GOSUB Z
3434 RAND N
3435 PRINT INT (RND*10)+1,INT (R
ND*10)+1
3437 GOSUB Z
3440 RAND O
3442 GOSUB Z
3445 REM WITH RAND SET TO ZERO
 RND WILL  RANDOMIZE
3447 GOSUB Z
3450 PRINT INT (RND*10)+1,INT (R
ND*10)+1
3452 GOSUB X
3455 CLS
3456 GOSUB L
3457 GOSUB X
3460 REM A MACHINE CODE CALL
3462 GOSUB X
3463 GOSUB G
3464 GOSUB U+T
3465 RAND USR 16514
3467 GOTO U
3500 LET B=K
3502 RETURN
```

```
3510 REM  REM / USR :REM IS A
 REMARK , USR IS A MACHINE CALL
3512 GOSUB X
3515 REM  REM REMINDS YOU WHAT IS
 IN THE LISTING
3517 GOSUB X
3520 REM   IT DOES NOT AFFECT THE
       PROGRAM
3522 GOSUB X
3525 REM TIMER ROUTINE (2.2 SEC)
3527 GOSUB X
3529 GOSUB K*S
3530 PAUSE 135
3531 GOSUB U-R
3532 GOSUB X
3535 REM  USR ROUTINES CAN BE
 CONTAINED IN A REM...
3537 GOSUB U-P
3538 GOSUB X
3540 REM E£RND▗(Y▚?Y?F(▗"")4 ?Y▚C
3542 GOSUB X
3543 GOSUB VAL "111"
3544 GOSUB U+T
3545 RAND USR 16514
3546 GOSUB VAL "111"
3547 GOSUB X
3548 GOSUB VAL "111"
3549 GOSUB U+T
3550 LET N=USR 16514
3560 GOTO U
3600 LET B=I
3602 RETURN
3610 REM  SAVE  USED FOR SAVING
 PROGRAMS ON TAPE
3612 GOSUB X
3615 REM  SAVE IS WRITTEN
   SAVE "  "   OR   SAVE "ANY WORD"
3616 PRINT AT 5+I,0;"( THIS IS A
 5 SEC DEMO SAVE )"
3617 GOSUB X
3618 GOTO VAL "3621"
3620 SAVE "SHORT TEST"
3621 RAND USR VAL "16643"
3622 SLOW
3623 GOSUB X
3625 REM  IF YOU SAVE A STRING
 BE SURE YOU GET THE SPELLING
 RIGHT WHEN YOU LOAD AGAIN
3627 GOSUB X
3630 REM  SAVE MAY BE ENTERED
 AS A DIRECT COMMAND
3650 GOTO U
3900 LET B=R
3902 RETURN
3910 REM  SCROLL MOVES DISPLAY
 UP ONE LINE
3912 GOSUB X
3915 FOR N=1 TO 5
3917 GOSUB X
3920 PRINT AT 20,0;"SCROLL DEMO"
3922 GOSUB X
3925 SCROLL
3929 GOSUB X
3930 SCROLL
3933 POKE E+R*K+I,T
3934 GOSUB X
3935 NEXT N
3936 LET A=VAL "3940"
3937 GOSUB U+T+S
3938 GOSUB X
3940 CLS
3941 LET A=VAL "3910"
3945 GOTO U
4100 LET B=R-O
4102 RETURN
4110 REM  STOP / RUN  USED TO
 STOP AND START A PROGRAM
4112 GOSUB X
4115 REM  STOP WILL STOP PROGRAM
 AT A GIVEN LINE
4117 GOSUB X
4118 POKE VAL "16418",0
4119 GOTO VAL "4121"
4120 STOP
4121 PRINT AT T,P;"(PRESS ""ENTE
R"" TO RESTART)"
4122 PRINT AT T+W,0;"9/4120"
4123 GOSUB L
4124 GOSUB X
4125 REM  CONT / GOTO / RUN
 WILL START PROGRAM AGAIN
4126 CLS
4127 GOSUB L
4129 GOSUB X
4130 REM  RUN WILL START PROGRAM
 AT SPECIFIED NO. OR VAR. LINE
4132 GOSUB X
4133 GOTO O
4135 RUN USB
4200 LET B=S+I
4202 RETURN
```

```
4210 REM  VAL/LEN/STR$/TO  USED
 IN STRING MANIPULATION
4212 GOSUB X
4215 LET A$="TS COMPUTERS"
4217 GOSUB X
 220 LET B$="TS 1000"
4222 GOSUB X
4225 PRINT A$,LEN A$
4227 GOSUB X
4230 PRINT B$,LEN B$
4232 GOSUB X
4235 PRINT A$(1 TO 2),A$(4 TO )
4237 GOSUB X
4240 PRINT B$(1 TO 2),B$(3 TO )
4242 GOSUB X
4245 REM  STR$ VAL ARE USED FOR
4247 GOSUB X
4250 REM  CHANGING STRINGS TO
 NUMBERS  NUMBERS TO STRINGS
4252 GOSUB X
4255 LET A$="1205"
4257 GOSUB X
4260 PRINT "A$=""";A$;""""
4262 GOSUB X
4265 PRINT A$+A$,VAL A$+VAL A$
4267 GOSUB X
4270 PRINT VAL A$(3 TO )*VAL A$(
3 TO )
4272 GOSUB X
4275 LET A$=STR$ 1000
4277 GOSUB X
4280 PRINT "A$ IS NOW """;A$;"""
"
4285 GOTO U
5200 LET B=R
5202 RETURN
5210 REM  SINCLAIR MATH CAN BE
 USED IN EXTENDED ALGEBRAIC FORM
 OR IN SIMPLE EXPRESSIONS
5212 GOSUB X
5215 PRINT 5*2+7,5*(2+7)
5217 GOSUB X
5220 PRINT 5*5*5,(10-5)*(25/5)*(
50R 25)
5222 GOSUB X
5225 PRINT 5**2,5**3
5227 GOSUB X
5230 PRINT 125/25/5,125/(25/5)
5232 GOSUB X
5235 PRINT PI,INT PI
5237 GOSUB X
5240 PRINT -((PEEK 16404+256*PEE
K 16405)-16509)+16384;" BYTES LE
FT"
5245 GOTO U
9001 SAVE "▗"
9002 RAND USR 16675 #=1 LOCK
9099 GOTO O
9500 REM  FAST/SLOW FILL DEMO
9505 PRINT
9510 FOR N=1 TO 64
9520 PRINT "▗";
9530 NEXT N
9550 RETURN
```

Line 9099 (invisible) contains jump to 16678 at 31574 (for version 2.2)

NEW REPORTS CLAIM QL IS PURE HELL SOMETIME

...Not Really Great But STILL A GOOD BUY

Ralph Coletti, who has published his last newsletter (due to illness), has always been one of the great forces in the American Sinclair Timex world. The Southeast Regional Newsletter that he published for three years has always had a mixture of old news and some of the most up-to-date material you could hope to find anywhere. The final issue has an excellent review of the QL which we will publish in our next issue. There are a lot of shortcomings and problems with the machine: it is by no means a great machine, by no means an exceptionally fast machine. It is still, says the author of the review, good value for your money--just not incredible or "awesome" as some teenage Yanks would have it.

HOW TO GET BACK ISSUES OF TSUG NEWSLETTERS?

4 Issues for $5.00
First Class Mail
Back to You...

PROGRAMS, REVIEWS, INFORMATION

by Charlie Day

Dear Mr. Dewey;

Hi! How are you? It has been along time since I've heard form you or TSUG. I haven't done anything wrong to get on forget this kids address list, have I ? Or is TSUG taking a summer break. After TIMEX left me out in the could you can see why I'm a little afraid of anything behind schedule. I've enclosed a review of the 2050 modem. You want to talk about world record grass mowing. I've done every blankity blank yard in Gaston County, but I got myself a modem. Hawg Wild software has fig-fourth for the 2068. I received a very informitve letter form Mr. Smith. The price for the Fig-Fourth for the 2068 is 29.95 plus 2.00 p&h. The news only gets better they also have the CHAMELEON EPROM for 55.00 plus 2.00 p&h along witha selection of Spectrum software. That is about it for me. I want to thank-you! again for fixing my computer. Keep up the GREAT work!

Hardware Review: 2050 Modem

System: T/S 2068; T/S 1500; T/S 1000 with T/S 1016

Price: $119.95

Manufacturer: Westridge Communications, Inc.,Marina Del Rey , Ca 90292 ; (213) 306-4103

Supplier: E Arthur Brown Comp. 3404 Pawnee Drive,Alexandria, Mn 56308: (612) 762-8847

The 2050 modem is the same modem that TIMEX planned to market, but it is not being soild under the TIMEX name. The 2050 is especially designed to work with all TIMEX computers so interfacing and operating is no problem. The 2050 comes with a informitve 27 page manual. The package also includes the power cable to the 2050, which is identical to the power plug for the T/S 1000. The 2050 is attractivly designed and fits very neatly under the 2040 printer. The back of the modem has a jack for your phone,(but a phone is not nessary)a jack for the power supply and the phone jack cord. The 2050 package also includes Terminal I software. On one side of the tape is the software for the T/S 1000;T/S 1500, the other side has the software for the 2068. The modem has a constant baud rate of 300. This is about 30 characters per second, or about 300 words per minute. Some modems require you to dial using a telephone, and then disconnect you phone. With the Terminal software you can dial the "phone" from your keyboard. The modem also has the exception-al feature of being able to answer the phone without anybody being present. With the software you can set the duplex for either full of half. Half duplex is usually used when communicate-ing with other home computers. full duplex is used with mainframe computers. Word size can be set for 5,6,7, or 8. The most common is 7. Some computers require that blank bits be sent after each character this is called stop bits. You can have one or two stop bits. One is the most common. Parity is used to determine if the words received contain errors or not. The 2050 can have even,odd,or no parity. Even is most common. All of these parameters can easily be changed from the Menu in the software. If you are connected with an information service all of these parameters are set for what is most common, so chances are you will not need to change any of them. If you do need to change the parameters it is as easy as pressing two keys.

Communicating with the 2050 is simple. First load the software,plug the power into the back of the modem. Next, if you are expecting a call from another computer owner you need only to enter the terminal mode so you can see the message. The modem will automaticaly answer the phone. If you want to call another computer, select the auto dial option,type the number in on the keyboard and press 'enter'. The computer will dial the number for you. If the line is busy or there is no answer the computer will inform you of this and return to the menu. If the other computer answers and sends it's carrier tone the screen will have the message CONNECT appear on it. The computer will automaticaly enter the terminal mode. You are now ready to communicate with the other computer. There are alots of other nice features in Terminal I software, for example you can print the screen display to the printer. The only option that I think would be nice if you could down-load and up-load data to and from tape. I'm happy to report that this limitation is in the software not the hardware. I can only hope that Micro-Systems Inc., the developer of the software does come out with the Terminal II software planned by TIMEX. I have been told that it has already been developed.

In summary I think that the 2050 modem is well worth the price. It is a simple to use modem and as far as I can tell it works fine with the T/S 2068 and T/S 1000. I have been unable to find the "bugs" that TIMEX claimed the modem had. I've called TIMEX to ask exactly what the so called "bugs" were, but they had no statement on the matter. If you get your own

2050 I will be more than happy to let you call me and test it out. My number is-(704)-865-8969. Happy telecommuications to all of you!!!

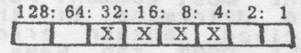
The UDGs Meet The Data Statement

By: Charlie Day

Accompanying this article is a simple but fun card game called "Acey-Ducy". The game of course is another masterpiece, and a silicon chips is a new snake food. Enough of the wisc cracks, the purpose of this article is to describe the technique used to create the eleven user defined graphics (UDG). The method, described in the T/S 2068 manual for creating a UDG , can be akward at times. The BIN command is a very useful command, but by entering characters in this style they can't easily be saved within a program.

To use the method I used to make the UDGs; simply make an 8 by 8 gride and in the squares darken the squares that you wish to have darkened. The top section of your grid might look like this:

128: 64: 32: 16: 8: 4: 2: 1



All we do now is add up any of the above numbers that is above a darkened square. In this example line we would have: 32+16+8+4= which of course equals 60. We would add up the values for the other same lines by using the same method. We would then set up a simple FOR,NEXT loop:

```
9000 FOR a=USR"a" TO USR "a"+7
9010 READ user : POKE a,user
9020 NEXT a
9030 DATA 60,.....
```

We would then place the added up end line numbers in the data statement. If we had two user defined graphic characters line 9000 would read:

```
9000 FOR a=USR "a" to USR "b"+7
```

Now type in my masterpiece and get to work on your own UDGs.

```
 20 GO SUB 9000
 30 LET d=50
 40 GO SUB 7000
 45 CLS
 50 LET a=INT (RND*13)+1
 60 LET b2=INT (RND*13)+1
 62 IF ABS (b2-a)<2 THEN GO TO
50
 65 LET g=INT (RND*13)+1
 66 IF a=d OR b2=d THEN GO TO 6
5
 70 PRINT AT 11,2;"┌──┐"
 71 LET b=INT (RND*4)
 72 LET c=0
 73 IF b=0 OR b=3 THEN LET c=2
 80 LET b$=CHR$ (144+b)
 90 PRINT AT 12,2;"|"; INK c;b$
; INK 0;" |"
 100 PRINT AT 13,2;"|  |"
 110 PRINT AT 14,2;"| ";(CHR$ 8
AND a>=10);a;" |"
 120 PRINT AT 15,2;"|  |"
 130 PRINT AT 16,2;"| "; INK c;
b$; INK 0;"|"
 140 PRINT AT 17,2;"└──┘"
 145 LET b=INT (RND*4)
 150 LET b$=CHR$ (144+b)
 151 LET c=0: IF b=0 OR b=3 THEN
 LET c=2
 160 PRINT AT 11,8;"┌──┐"
 170 PRINT AT 12,8;"|"; INK c;b$
; INK 0;" |"
 180 PRINT AT 13,8;"|  |"
 190 PRINT AT 14,8;"| ";(CHR$ 8
AND b2>=10);b2;" |"
 200 PRINT AT 15,8;"|  |"
 210 PRINT AT 16,8;"| "; INK c;
b$; INK 0;"|"
 220 PRINT AT 17,8;"└──┘"
 230 PRINT AT 1,5;"You have $";d
 240 PRINT AT 3,3;"How much do y
ou bet my next"'TAB 3;"card lies
 between ";a;" and ";b2;"?"
 250 INPUT e
 260 IF e>d THEN GO TO 150
 270 IF e<1 THEN PRINT 'TAB 10;
FLASH 1;"Coward"
 280 PRINT 'TAB 10;"My number is
 ";g
 290 IF e<1 THEN GO TO 3000
 300 IF NOT (g>a AND g<b2 OR g<a
 AND g>b2) THEN GO TO 350
 310 PRINT 'TAB 5;"Well done, yo
u win $";2*e
 320 LET d=d+2*e
 330 GO TO 3000
 350 PRINT 'TAB 9;"Sorry, you lo
se $";e
 360 LET d=d-e
 370 IF d<1 THEN PRINT "You are
broke": GO TO 3020
3000 PRINT AT 20,10;"Press any k
ey": PAUSE 0
3010 GO TO 45
3020 INPUT "Press enter to play
 again "; LINE a$: RUN
7000 BORDER 0: PAPER 4: INK 9: C
LS
7990 RETURN
9000 FOR a=USR "a" TO USR "k"+7
9010 READ user: POKE a,user
9020 NEXT a: RETURN
9030 DATA 24,60,126,255,255,126,
60,24,24,60,90,255,255,90,24,60,
16,56,124,254,254,64,16,56
9060 DATA 68,238,254,254,254,124
,56,16,204,204,51,51,204,204,51,
51,0,0,0,7,15,12,24,24
9090 DATA 0,0,0,192,240,48,24,24
,24,24,12,15,7,0,0,0,24,24,48,24
0,192,0,0,0
9120 DATA 0,0,0,255,255,0,0,0,24
,24,24,24,24,24,24,24
9990 REM a b c d e f g h i j k
9991 REM ♦ ♣ ♠ ♥ ▒ ╱ ╲ ╳ ═ ▏
9998 SAVE "acey" LINE 0
9999 VERIFY "acey"
```

Smart Mailbox

By: Charlie Day'

Have a little time on your hands? Typing this in should keep you busy. Smart Mailbox is a simple mailing list program. I've had a dislike of arrays for sometime, this is basicly due to their wasted memory space. To be able to perform the same task as an array without the wasted memory, I use this long string method. You have probably seen this method used before.

```
  10 LET a$=CHR$ 1: LET p=0
 100 PAPER 7: CLS : INK 4: BORDE
R 1
 120 PRINT AT 1,10;"Smart Mailbo
x"
 130 PRINT AT 4,5;1;TAB 10;"To E
nter Address";AT 6,5;2;TAB 10;"T
o Delete Address";AT 8,5;3;TAB 1
0;"To Edit Adress";AT 10,5;4;TAB
 10;"To Search for Address";AT 1
2,5;5;TAB 10;"To Save Data";AT 1
4,5;"6";TAB 10;"To Load Data"
 210 PRINT FLASH 1;AT 20,6;" Ent
er One Of Above"
 220 LET q$=INKEY$
 230 IF CODE q$<49 OR CODE q$>54
 THEN GO TO 210
 250 CLS
 260 GO TO VAL q$*1000
1010 CLS : PRINT '"Enter Full Na
me? ";: INPUT b$: PRINT INK 0'b$
'
1050 IF b$="" THEN GO TO 100
1060 LET a$=a$+b$+CHR$ 24
1070 PRINT ,,"Enter Street Addre
ss ? ";: INPUT B$: PRINT INK 0'b
$'
1090 LET a$=a$+b$+CHR$ 25
1100 PRINT ,,"Enter City ? ";: I
NPUT B$: PRINT INK 0;b$
1120 LET a$=a$+b$+CHR$ 26
1130 PRINT ,,"Enter State ? ";:
INPUT B$: PRINT INK 0;b$
1150 LET a$=a$+b$+CHR$ 27
1400 PRINT ,,"Enter Zip ? ";: IN
PUT B$: PRINT INK 0;b$
1416 PRINT '"Press any key to co
ntinue": PAUSE 0
1420 LET a$=a$+b$+CHR$ 1
1430 GO TO 1000
2010 PRINT AT 10,0;"Enter Keywor
d of Data to be      Deleted:::"
2030 INPUT c$
2050 FOR n=1 TO LEN a$-LEN c$+1:
 IF a$(n TO n+LEN c$-1)=c$ THEN
GO TO 2200
2070 NEXT n
2080 CLS : PRINT AT 10,0;"DATA N
OT IN FILE:::"'''"PRESS ANY KEY
TO CONTINUE": PAUSE 0
2090 CLS : GO TO 100
2200 FOR a=n TO 1 STEP -1
2210 IF CODE a$(a)=1 THEN GO TO
2230
2220 NEXT a
2230 FOR b=n TO LEN a$
2240 IF CODE a$(b)=1 THEN GO TO
2260
2250 NEXT b
2260 CLS
2270 PRINT 'a$(a+1 TO b-1)
2275 RESTORE
2280 PRINT '"IS THIS THE DATA BL
OCK YOU WANT DELETED?": PAUSE 0
2300 IF INKEY$="N" OR INKEY$="n"
 THEN GO TO 2070
2310 LET a$=a$( TO a)+a$(b+1 TO
LEN a$)
2320 GO TO 100
3010 PRINT '"Enter Keyword?": I
NPUT c$
3050 FOR n=1 TO LEN a$-LEN c$+1
3060 IF a$(n TO n+LEN c$-1)=c$ T
HEN GO TO 3100
3070 NEXT n
3080 GO TO 2080
3100 FOR a=n TO 1 STEP -1
3120 IF CODE a$(a)=1 THEN GO TO
3140
3130 NEXT a
3140 FOR b=n TO LEN a$
3150 IF CODE a$(b)=1 THEN GO TO
3170
3160 NEXT b
3170 LET d$=a$(a+1 TO b-1)
3175 GO TO 3800
3180 FOR i=1 TO LEN d$
3190 IF CODE d$(i)=24 THEN GO TO
 3210
3200 NEXT i
3210 CLS
3220 PRINT '"Full Name is-";d$(
TO i-1)
3230 PRINT '"Enter new name or p
ress enter   for no change?"
3250 INPUT e$
3260 IF e$="" THEN LET e$=d$( TO
 i-1)
3265 LET d$=e$+d$(i TO LEN d$)
3275 LET f$=e$+CHR$ 24
3280 CLS
3290 LET c=i+1
3300 FOR i=c TO LEN d$
3310 IF CODE d$(i)=25 THEN GO TO
 3330
3320 NEXT i
3330 PRINT '"Street address is-
";d$(c TO i-1)
3340 PRINT '"Enter new address o
r press enter   for no change?"
3360 INPUT e$
3370 IF e$="" THEN LET e$=d$(c T
O i-1)
3380 LET f$=f$+e$+CHR$ 25
3400 CLS
3410 LET c=i+1
3420 FOR i=c TO LEN d$
3430 IF CODE d$(i)=26 THEN GO TO
 3450
3440 NEXT i
3450 PRINT '"City is-";d$(c TO i
-1)
3460 PRINT '"Enter new city or p
ress enter   for no change?"
3470 INPUT e$
3480 IF e$="" THEN LET e$=d$(c T
O i-1)
3490 LET f$=f$+e$+CHR$ 26
3510 CLS
3520 LET c=i+1
3530 FOR i=c TO LEN d$
3540 IF CODE d$(i)=27 THEN GO TO
 3560
3550 NEXT i
3560 PRINT '"State is-";d$(c TO
i-1)
3565 PRINT '"Enter new state or
press enter   for no change?"
3580 INPUT e$
3590 IF e$="" THEN LET e$=d$(c T
O i-1)
3600 LET f$=f$+e$+CHR$ 27
3620 CLS
3630 LET c=i+1
3640 PRINT '"Zip Code is-";d$(c
TO LEN d$)
3650 PRINT '"Enter new Zip Code
or press enter   for no change?"
3660 INPUT e$
3670 IF e$="" THEN LET e$=d$(c T
O LEN d$)
3680 LET f$=f$+e$
3700 CLS
3710 LET a$=a$( TO a)+f$+a$(b TO
 LEN a$)
3720 GO TO 100
3800 CLS
3810 PRINT INK 0;d$
3820 PRINT '''"Is this the corret
 data block?"
3830 PAUSE 0
3840 IF INKEY$="Y" OR INKEY$="y"
 THEN GO TO 3180
3850 GO TO 3070
4010 CLS
4020 PRINT AT 10,0;"Enter Keywor
d of Phrase?"
4040 INPUT c$: CLS
4070 FOR n=1 TO LEN a$-LEN c$+1
4080 IF a$(n TO n+LEN c$-1)=c$ T
HEN GO TO 4200
4090 NEXT n
4100 PRINT AT 10,0;"Data not in
file:::Please check keyword"
4120 PRINT '''"Press any key to c
ontinue": PAUSE 0: GO TO 100
4200 LET nn=n
4205 FOR a=n TO 1 STEP -1
4210 IF a$(a)=CHR$ 1 THEN GO TO
4230
4220 NEXT a
4230 FOR b=n TO LEN a$
4240 IF a$(b)=CHR$ 1 THEN GO TO
4300
4250 NEXT b
4300 LET d$=a$(a TO b)
4310 FOR n=1 TO LEN d$
4320 IF CODE d$(n)=24 THEN GO TO
 4335
4330 NEXT n
4335 IF p=1 THEN GO SUB 4850
4340 GO SUB 4800
4350 FOR n=1 TO LEN d$
4360 IF CODE d$(n)=25 THEN GO TO
 4375
4370 NEXT n
4375 IF p=1 THEN GO SUB 4850
4380 GO SUB 4800
4390 FOR n=1 TO LEN d$
4400 IF CODE d$(n)=26 THEN GO TO
 4415
4410 NEXT n
4415 IF p=1 THEN GO SUB 4860
4420 GO SUB 4840
4430 FOR n=1 TO LEN d$
```

```
4440 IF CODE d$(n)=27 THEN GO TO
 4455
4450 NEXT n
4455 IF p=1 THEN GO SUB 4860
4460 GO SUB 4840
4465 IF p=1 THEN LPRINT d$(2 TO
LEN d$-1): GO TO 4710
4700 PRINT INK 0;d$(2 TO LEN d$-
1)
4710 PRINT '"Press 'P' to pr
int label toprinter"'"Press s
pace bar to return to Menu"
4711 PRINT '"Press any other key
 to continue search": PAUSE 0
4712 CLS
4715 IF INKEY$="P" OR INKEY$="p"
 THEN LET p=1: GO TO 4070
4716 LET p=0
4720 IF INKEY$=" " THEN GO TO 10
0
4730 FOR n=nn+1 TO LEN a$-LEN c$
+1
4780 GO TO 4080
4799 STOP
4800 PRINT INK 0;d$(2 TO n-1)
```

Super Variable

By Charlie Day

I've been a serious programer for about three years, and to this day, I've never learned to like arrays. My main dislike of arrays is the wasted memory space sometimes encountered. But what could I do, arrays are very powerful tools. My answer was to use string variables instead of arrays. Now, wait before you get all excited, I know a string variable cannot be as powerful as an array but it can be used to save memory. Let me demonstrate:

```
10 PRINT "Enter first name"
20 INPUT F$
30 PRINT "Enter middle name"
40 INPUT M$
50 PRINT "Enter last name"
60 INPUT L$
70 LET A$=F$+M$+L$
80 PRINT A$
```

I know what you are thinking "That is great, now how are you going to seperate the individual data items." How about if we placed something before and after each piece of data. If we did this we could slice the string and read each character one at a time. When we encountered the something we placed between the data we would read on until we found the next something, then we would know everything inbetween would be the data wanted. That is going to be a little easier said than done. The first problem is what do we place in the main string to seperate the data. We can not use just any character. The reason for needing a special character is we do not want to use a character that might be part of the main string. For the answer to the special something needed try the below programs.

T/S 1000

```
1 FOR I=67 TO 111
2 PRINT CHR$ I
3 NEXT I
```

For the T/S 2068 change line 1

```
1 FOR I=24 TO 31
```

Now run the program. What do you mean all you see is question marks. Those are unused characters. But these are characters just the same. You will not run into the problem of these characters being part of your main string. Delete lines one through three. Change line

```
4820 LET d$=d$(n TO LEN d$)
4830 RETURN
4840 PRINT INK 0;d$(2 TO n-1);"
";: LET d$=d$(n TO LEN d$): RETU
RN
4850 LPRINT d$(2 TO n-1): RETURN

4860 LPRINT d$(2 TO n-1);" ";: R
ETURN
5010 INPUT "File name ?";n$
5040 PRINT "Saving a:  ";n$
5045 LET a=LEN a$: DIM b$(a): LE
T a$=a$+" "
5050 FOR n=1 TO a: LET b$(n)=a$(
n TO n+1): NEXT n
5060 SAVE n$ DATA b$()
5090 GO TO 100
6005 CLEAR
6010 INPUT "Filename ?";n$
6020 LOAD n$ DATA b$()
6025 LET i=LEN b$(): PRINT i: ST
OP
6030 LET a$=b$()
6050 GO TO 100
7000 SAVE "Mail" LINE 10
```

seventy to: LET A$=F$+CHR$ 67+M$+CHR$ 67+L$+CHR$ 67; for the T/S 1000; change the CHR$ 67 to CHR$ 24 for the T/S 2068.

Now run the program. Alright, so it does not do what I said it would do. I'm not finished yet. Add the following lines to your program.

```
80 CLS
90 LET C=1
100 FOR I=1 TO 3
110 FOR N=C TO LEN A$
120  IF A$(N)=CHR$ 67 THEN GOTO
 140
130 NEXT N
140 PRINT A$(C TO N-1)
150 LET C=N+1
160 NEXT I
```

On the 2068 change CHR$ 67 in line 120 to CHR$ 24. Now run the program.

The variables are now neatly seperated from the main variable A$. This technique is great in programs that require large amounts of data, of couse it does have it's draw backs. But it does conserve memory.

```
 1 REM This program will tell
the day of the week that a date
will fall on. Enter all prompts
in numberical form. Ex: 10/5/198
4
 100 LET a$="  MonTueWedThuFriSa
tSun"
 110 INPUT "What is the date of
the day? ";d: PRINT "Day",d
 115 LET d$=STR$ d: IF LEN d$>2
THEN GO TO 110
 120 INPUT "Enter month that dat
e of the day is in ";m: PRINT "M
onth",m
 125 LET m$=STR$ m: IF LEN m$>2
THEN GO TO 120
 130 INPUT "Enter the year (in f
ull: ex 1984)";y: PRINT "Year",y

 140 LET y$=STR$ y: IF LEN y$<4
THEN GO TO 130
 150 LET q=y-(m<3)
 160 LET k=q/100
 170 LET t=m+12*(m<3)
 180 LET r=INT (13*(t+1)/5)+INT
(5*q/4)-INT k+INT (k/4)+d+5
 190 LET r=r-(7*INT (r/7))+1
 200 PRINT d;"/";m;"/";y;"-";a$
(r*3 TO r*3+2)
 210 PRINT "Press any key to re
turn": PAUSE 0: GO TO 100
```

ONE MORE PROGRAM FROM CHARLIE DAY

16K T/S 1000

```
 1 REM ACEY-DUCEY BY CHARLIE
DAY
 2 LET D=20
 3 FAST
 4 PRINT "
"
 5 FOR I=1 TO 19
 6 PRINT "
"
 7 NEXT I
 8 PRINT "
"
 30 LET A=INT (RND*13)+1
 40 LET B=INT (RND*13)+1
 50 IF ABS (B-A)<2 THEN GOTO 30
 60 LET C=INT (RND*13)+1
 70 IF A=C OR B=C THEN GOTO 60
 80 PRINT AT 11,2;"   ";AT 12,
2;"   ";AT 13,2;"   ";(CHR$ 0 AND
 A<=9);A;" ";AT 14,2;"   ";AT 1
5,2;"   "
 90 PRINT AT 11,8;"   ";AT 12,
8;"   ";AT 13,8;"   ";(CHR$ 0 AND
 B<=9);B;" ";AT 14,8;"   ";AT 1
5,8;"   "
 100 PRINT AT 1,5;"YOU HAVE $";D
 110 PRINT AT 3,1;"HOW MUCH DO Y
OU BET MY NEXT";: NUMBER LIES BE
TWEEN ";A;" AND ";B;"?"
 120 SLOW
 150 INPUT E
 160 IF E>D THEN GOTO 150
 170 IF E<1 THEN PRINT AT 5,5;"C
OWARD"
 180 PRINT AT 7,5;"MY NUMBER IS
";C
 200 IF E<1 THEN GOTO 280
 210 IF NOT (C>A AND C<B OR C<A
AND C>B) THEN GOTO 250
 220 PRINT AT 9,2;"WELL DONE YOU
WIN $";2*E
 230 LET D=D+2*E
 240 GOTO 280
 250 PRINT AT 9,2;"SORRY,YOU LO
SE $";E
 260 LET D=D-E
 270 IF D<1 THEN PRINT AT 10,2;"
YOU ARE BROKE"
 275 IF D<1 THEN STOP
 280 PRINT AT 18,10;"PRESS ANY K
EY"
 285 PRINT AT 18,10;"PRESS ANY K
E"
 290 IF INKEY$="" THEN GOTO 280
 300 CLS
 310 GOTO 8
 990 SAVE "ACE"
 999 RUN
```

PSEUDO ROM BOARD
by Hiawatha Demby

Hi Demby has developed a nice $20 to $30 addition to your TSZX computer. We will be reviewing it in the next issue. In the meantime, here are some programs for those of you who have purchased a board recently.

PSEUDO ROM PROGRAMS

THE FOLLOWING PROGRAMS WILL ALLOW YOU TO CREATE AND SAVE THE CHARCTER SETS YOU CREATE ON YOUR PSEUDO-ROM BOARD. THEY FIX AND RUN IN A 2K COMPUTER. NOTE THAT THE SET SAVER ONLY SAVES THE GRAPHICS AREA OF THE CHARCTERS. A BLOCK MOVE OF THE OTHER CHARCTERS BEFORE RELOADING SETSAVER WILL PRODUCE A READIBLE SCREEN.

TINY CHARCTER MAKER

```
 1 REM Y4 GOSUB ?TAN
 2 LET START=8192
 10 PRINT "NEW CHARTER SET?"
 20 INPUT A$
 30 IF A$="YES" THEN GOTO 50
 40 GOTO 100
 50 PRINT "NEW SET STARTS AT?"
 52 INPUT START
 54 FAST
 56 FOR X=0 TO 511
 60 POKE (START+X),PEEK (7680+X
)
 70 NEXT X
 80 RAND USR 16514
 82 SLOW
 100 PRINT AT 0,0;"   WHAT CHARC
TER?";AT 21,0;"                "
 110 INPUT C$
 114 IF C$="" THEN STOP
 116 PRINT AT 10,10;C$
 120 LET CADD=(CODE C$)*8+START
 130 FOR X=0 TO 7
 140 PRINT AT 21,0;"LINE ";X+1
 150 INPUT D$
 152 IF D$="" THEN GOTO 170
 160 POKE CADD+X,VAL D$
 170 NEXT X
 180 PRINT AT 0,0;"
"
 190 GOTO 100
```

```
 1 REM Y4 GOSUB ?TAN XXXXXXXXX
XXXXXXXXXX
1000 PRINT "SET LOCATION?"
1010 CLEAR
1020 INPUT L
1022 FAST
1030 CLS
1032 DIM A(256)
1040 FOR X=1 TO 256
1050 LET A(X)=PEEK L
1060 LET L=L+1
1070 NEXT X
1080 PRINT "START RECORER."
1090 INPUT Q
1100 SAVE "SET819"
1110 LET L=8192
1120 FOR X=1 TO 256
1130 POKE L,A(X)
1140 LET L=L+1
1150 NEXT X
1160 RAND USR 16514
1170 CLEAR
1172 SLOW
1180 STOP
```

Dear T/S Friend,

Due to an oversupply of certain issues, TIMELINEZ is making the following offer:

TIMELINEZ, the joint publication of three T/S groups in the San Francisco/Oakland/San Jose areas will send free copies of the March and May issues upon request. Please send a legal size SASE containing 2 stamps.

===

TIMELINEZ
P.O. BOX 1312
PACIFICA, CA 94044

Feel free to convey this offer to your fellow user group members.

ZEBRA SYSTEMS INC

78-06 Jamaica Avenue, Woodhaven NY 11421 (212) 296-2385

NEW ZEBRA SYSTEMS BULLETIN BOARD FOR TIMEX / Sinclair USERS

(212) 296-2229

Zebra Systems, Inc. is proud to announce that the Zebra Systems BBS is now on-line and available To All Timex Modem Users,

The Zebra Systems BBS is devoted to Timex/Sinclair users nationwide. Available 24 hours a day, 7 days a week it will be a central information exchange for Timex / Sinclair Computer owners across the U.S.A. Access to the BBS will be free of charge (Only telephone company charges apply.), we do not charge for connect-time.

Features include:

* General Message Base for conversing with Users Nation-wide.

* SIGS (Special Interest Groups) pertaining to the TS-1000 & TS-2068.

* E-Mail Section for User-to-User private mail.

* Zebra Systems Shop-at-Home service for Timex/Sinclair related products.

* Up-To-Date News on Timex happenings, new products, product reviews, etc...

Zebra Systems BBS is on line and ready for your call at

(212) 296-2229.

WELCOME NEW USER!!!
==========

. Welcome to Zebra Systems BBS!

. This system is the first BBS in the New York Metropolitan area devoted exclusively to TIMEX/Sinclair computers and their users.

. We hope to keep you up to date with new developments, products, news, product reviews and to be a central point of contact between you and other users of TIMEX/Sinclair equipment.

. Feel free to ask questions of the other users about hardware, software, technical questions, or just say hi! You will find Special Interest Groups (SIGs) in the Timex Sub-board section (Selection #2 in the MAIN MENU). In the message bases accessed from that menu you may converse about your particular equipment in the appropriate sub-board.

. Many interesting TIMEX/Sinclair related products are available for purchase in the <M>erchandise section. Before you purchase any product why don't you see if it's listed,and if so what our price is. This BBS is supported by sales of TIMEX/Sinclair P roducts by ZEBRA SYSTEMS, INC. If you see any products you feel are useful to TIMEX/Sinclair users, please leave a Message to the Sysop describing the item, it's price, who manufactures it, and in which publication you saw the ad.

. Our growing Club section (Selection #4 from the Main Menu...) will contain listings of Timex clubs nation-wide, and message bases for those who request them. If you know of any groups or clubs NOT listed please leave a message to the Sysop giving the name, mailing address, and a contact (preferably an officer) of that club.

. Please use a password when logging on, this password insures that your private mail can only be read by you and you ONLY! Please do not give another user you password under any circumstances. If you forget your password, contact the Sysop in order to change or re-obtain your password. Please use a different password on this system than other BBS systems.You may change your Password and terminal specifications by accessing the <U>tility section from the MAIN MENU. (REMEMBER: Upper & Lower case letters are treated as different characters, so please remember whether your password was entered in Upper or Lower case.)

. The one request we make of you in exchange for the priviledge of using this system is that you respect the following policies reguarding it's use:

1) Please no cursing or foul language.
(This is a family BBS, and we Timex users know better words than that anyway...)
2) ABSOLUTELY NO PIRATING OR TRADING MESSAGES!!!!
3) NO PHONE PHREAKING OR OTHER SERVICE ABUSE MESSAGES!!!
(Both offenses are punishable by having access to this system permanently denied!)
4) Please use your real name... Fake or obviously phony names will be purged from the Userlog along with any messages left using that name IMMEDIATELY!!!

. Your compliance to these rules will make your use of this system pleasant for all, and allow it's continued existance as a Public "FREE" service...

PROGRAMMER'S UTILITY EPROM
by BILL ROBERTS

About two weeks ago I received the Programmer's Utility EPROM from Delphic Enterprises, P.O. Box 72205, Corpus Chirti, Tx 78472. I received the EPROM about three weeks after mailing the check (which I think is fairly quick turnaround). While I do not feel that I have spent nearly enough time using the EPROM, I wanted to share with you my initial impressions.

With some trepidation, the EPROM was installed in the left socket (component side) of a CAI/O board. A word of caution--if a CAI/O board is used, and a stringy floppy EPROM is also installed, the left socket is the only one which will work with the EPROM. As set up, the 8K to 10K block of memory is used by the EPROM. When the computer was powered up and USR 8192 entered, the menu actually came up. I have tested all of the routines (except for merge and unlock) and they work great.

The eight routines are: Renumber, Copy, Search, Delete, Merge, Rem Kill, Free Space, and Unlock. In effect the TSZX can now be expanded to a 10K operating system. For example a section of code can now be EASILY moved by using the Copy routine followed by deleting the old section of code. Another very useful item is the Merge routine. Although it is possible to merge two programes using the 8K ROM, it is painful. With the Merge routine on EPROM however, it becomes much easier--almost routine.

The shining star of the EPROM is the Renumber routine. I do not know how many times I have had to renumber statements as a program developed. Using the renumber routine makes the job almost a real joy.

There is one item to be aware of--if "the edit line pointer is placed on a nonexistent line at the end of a program"... use of the Copy routine will cause unpredictable results. I feel the EPROM programs are very well written and the documentation is also good. I give the Programmer's Utility EPROM a solid 4.

MOUNTAINEER SOFTWARE
115 NORTH 7th AVENUE PADEN CITY, WV 26159
(304) 337-8502

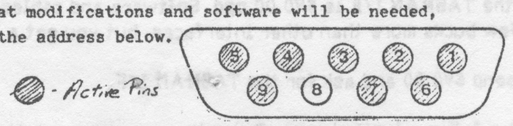
GREETINGS FROM "THE MOUNTAIN STATE"

My name is Bill Ferrebee, and I run MOUNTAINEER SOFTWARE in Paden City, WV. I would like to thank Douglass Dewey for the oppertunity to join this great T/S users group, and to be able to share ideas between each other. It is this kind of unity that will keep our cause alive.

In the past few weeks, some very exciting things have happened for me. First, I just worked out a deal to write reviews and a monthly column on TELECOMMUNICATIONS for T-S HORIZIONS magazine. Rick Duncan has a great publication, and I hope that you will look for the articles.

Speaking of telecommunications, there is a BBS here in West Virginia that will soon gear an entire board specifically to T/S owners and users. The TRADE-80 board at (304)652-1416 is currently general use, but the Sysop just procured a 5-Mg Hard Disk drive, and by August 1, we will have our own board for a T/S SIG. Call now and register yourself. Leave me a message when you call. I hope to keep a fully stocked Bulletin section and plenty of programs to download on the board.

Finally, I have a challange for someone to solve. I recently purchased a 12-key keypad for an Atari 2600. I would like to convert it to use on my T/S 2068 to enter numbers. It uses the standard joystick plug, and I have enclosed a diagram of the active pins. If you know what modifications and software will be needed, please contact me at the address below.



- Active Pins

In closing, again I would like to express how happy I am to be a part of this group. Please drop me a line or leave me a message on the BBS. Let's band together to further the Timex/Sinclair!

Pro/file Updates

The Newsletter For ZX Pro/file Users

Vol. 1, No. 3 — July 1984

AFTER THE FALL....

Bill Jones of Panama City, FL sent an interesting note. He says, "UPDATES has the potential of being a National Newsletter for Sinclair Computer Information. Looking further down the road, UPDATES could be a vehicle to start up a National Sinclair Computer Club."

Funny you should mention that, Bill. While UPDATES is definately not ever going to be a "general interest" sort of rag, I do see the need for some other publication to help fill the void left by SYNC cutting out. I am doing what I can to help out in this. Namely, I am publishing another newsletter devoted to a broad range of ZX/TS interests.

This newsletter is called THE SYNCWARE NEWS, a bimonthly journal for all Sinclair/Timex computer fanatics. Actually, SYNCWARE NEWS is not new. It has been around for the past year. What is new is my involvement in the publishing aspects of the journal. I hope to contribute whatever talent I have toward improving the format of SYNCWARE NEWS, broadening its scope, and boosting its circulation.

Since January we saw the Timex computer revolution take on all the characteristics of a classic Greek tragedy. Overnight we watched our sources of supplies and information dry up. It seems to me that the only people who didn't desert the computer were the ZX/TS owners themselves, and they (make that "we") were left standing saying, "Gee, what happened?"

Well I for one, firmly believe there is a whole lot more to be said about our lowly computers. SYNCWARE NEWS has assembled a huge collection of information about all ZX/TS machines which shall be published over the coming months. The complete spectrum of topics are covered: hardware projects, programming tips, product reviews, announcements, news, program listings, and tutorials on programming theory. We have something for every level of expertise--from rank amateur to cosmic.

To use the words of Bill Jones, SYNCWARE NEWS is "The Newsletter for Sinclair Information". We will do our utmost to keep the ZX/TS flame alive. But to insure success, we need subscribers. Now for the infamous bottom line:

For a one year subscription to SYNCWARE NEWS, send $16.95 (Visa/MC welcome) to:

SYNCWARE NEWS, Subscription Office
P.O. Box 64
Jefferson, NH 03583

Tell your friends about SYNCWARE NEWS and ask them to subscribe too. With a large active lively readership, SYNCWARE NEWS will insure that our marvelous little machines will be around for quite some time to come.

NEW for the TS2068... the TASMAN centronics printer interface

If you need an interface for your full size printer, I can now supply you with the TASMAN centronics I/F. This unit is made to be used with the TS2068 computer. The I/F comes in a case about 2 x 3 x 1 inch deep which plugs tightly into the rear edge connector. A program cassette comes with the I/F that allows you to use LPRINT and LLIST commands to make the printer work. A screen COPY command is also available which is implemented by means of a RANDOMIZE USR call.

There are several advantages to using the TASMAN I/F. First, it comes in a case. This makes it neat in appearance and less prone to damage. Second, the print driver software that accompanies the interface is very flexible and allows you to use just about any brand of parallel printer. There are actually 6 different programs on the cassette that allow for screen copies of text, HI-resolution graphics, even color if your printer can handle it. This is in addition to the standard LPRINT and LLIST functions. Third, the interface hardware consists of a 6821 PIA chip. If you're interested in learning how to write IN/OUT port programming of you're own, the TASMAN interface gives you the advantage of working with a programmable interface adapter (PIA). That is, 2 parallel output ports and 2 parallel inputs all in one package.

Price for the TASMAN I/F is $90.00 ppd. Software and cables are included FREE. That's a few bucks more than other interfaces, but you get a lot more too.

To order send $90.00 and ask for the TASMAN I/F.

Special For a limited time. Order the prowriter printer at the regular price and get the TASMAN printer interface for $30.00 OFF. That's a discount of 30 PERCENT!

Thomas B. Woods
P.O. BOX 64, JEFFERSON, NH 03583
Phone (603) 586-7734

NEW LIGHT ON THE MEMOTECH MEMORY MYSTERY

In UPDATES No. 1 I noted that several people experienced unusual behavior of their computers when the Memotech 16K ram pack was used to expand capacity to 32K.

This snowball turned into an avalanche as others added extra memory and other interfaces--particularly when a TS1500 was used as the base computer.

Bill Russell of Winky Board fame sheds some light on this common problem and gives a possible cure. Russell says:

One often overlooked cause of crashes in TS computers--especially when extra devices are plugged in or when a program is being run that uses virtually all available memory (like Pro/File does) is in the Z80 microprocessor itself. Timex computers employ a Z80 chip that is of such marginal quality that any extra load on its data or address lines simply cannot be handled.

If you run Pro/File and it crashes frequently for no apparant reason, try replacing the on board Z80 with one made by Intel or Mostek. I tried this and it ended almost all of my crashing problems; even many that I thought were caused by ram pack wobble.

```
 70 LET F=E+32*M
 80 FOR P=0 TO 7
 90 LET G=F+256*P
100 FOR Q=0 TO 31
110 LPRINT CHR$ PEEK (G+Q);
120 NEXT Q
130 NEXT P
140 NEXT M
150 NEXT N: STOP
```

```
  1>REM The Rowland's
  5 REM Olivetti PR2300
  6 REM Screen Dump
  7 REM "INVERSE IMAGE"
 10 CLS : LIST 1
 15 POKE 64256,0
 20 LET D=16384
 23 LPRINT CHR$ 27;CHR$ 45;
 25 LPRINT CHR$ 27;CHR$ 47;
 30 LPRINT CHR$ 27;CHR$ 71;"250
,032;024";CHR$ 27;CHR$ 90;
 40 FOR N=0 TO 2
 50 LET E=D+2048*N
 60 FOR M=0 TO 7
 70 LET F=E+32*M
 80 FOR P=0 TO 7
 90 LET G=F+256*P
100 FOR Q=0 TO 31
110 LPRINT CHR$ PEEK (G+Q);
120 NEXT Q
130 NEXT P
150 NEXT M: NEXT N: STOP
```

```
 10 CLS : LIST 1
 15 POKE 64256,0
 20 LET D=16384
 25 LPRINT CHR$ 27;CHR$ 47;
 30 LPRINT CHR$ 27;CHR$ 71;"250
,032;024;5";CHR$ 27;CHR$ 90;
 40 FOR N=0 TO 2
 50 LET E=D+2048*N
 60 FOR M=0 TO 7
 70 LET F=E+32*M
 80 FOR P=0 TO 7
 90 LET G=F+256*P
100 FOR Q=0 TO 31
110 LPRINT CHR$ PEEK (G+Q);
120 NEXT Q
130 NEXT P
150 NEXT M: NEXT N: STOP
```

```
1>REM The Rowland's
5 REM Olivetti PR2300
6 REM Screen Dump
7 REM "Tall Words"
```

To use, load in the AERCO driver, delete 1,9999, type in the program, and RUN.

```
*** South Bay Computer Club ***
* Time Sinclair User's Group *
```

Hi Douglass,
Thanks for the informative newsletter.

We have no problems out here with our 2068's.

Some of the members were 'griping' about the color, but after checking the units out against my 13" Magnavox "cheap" portable TV, everything looked great. Another good set is a 13" Gold Star. Stay away from electronic tuners and monitors, 'try before buying'.

I have inclosed a routine that might help your members make machine code program back-ups.

Keep in touch.

The next meeting will be held with the West Side ZX Group on Sat May 19, 1984 at 10:30 AM.

The location is:
Santa Monica Library -
Fairview Branch
2101 Ocean Park Blvd.

John W. Petersen (213)-545-9581
Apr 20, 1984

See the Next Page For this ROUTINE And General Method

*** NON-BREAKABLE PROGRAMS ***
2068 Back-Up Copies

1. MERGE "" - Stops Basic program.

2. Add STOP statements

3. Isolate the call-up command.

4. Finish LOADing program with commands in the Immediate mode.

5. Add SAVE statement - SCREEN$ are optional.

6. Try 31000 for CODE reserved space. If program crashes during LOADing, increase size in increments of 1000.

******* MEAT-AX APPROACH *******

John Petersen (213)-545-9581
Apr 15, 1984

BEFORE

```
  5 CLEAR 30000
 10 BORDER 1: PAPER 1: INK 1: C
LS
 20 POKE 23659,0: PRINT AT 22,0
;;
 25 FOR x=23232 TO 23263: POKE
x,9: NEXT x
 30 LOAD "s"CODE 16384
 40 POKE 23659,0: PRINT AT 22,0
;;
 50 LOAD "p"CODE
 60 PRINT USR 32768
```

AFTER

```
  5 CLEAR 30000
 10 BORDER 1: PAPER 1: INK 1: C
LS
 25 FOR x=23232 TO 23263: POKE
x,9: NEXT x
 50 LOAD "p"CODE
 59 STOP
 60 REM PRINT USR 32768
 69 STOP
 80 SAVE "p" LINE 5: SAVE "p"CO
DE 30001,32000
 99 STOP
```

USER FRIENDLY PROGRAMS FOR THE BEGINNER

By DON WEISS
27 Mitchell St.
Randolph, Ma 02368

This is intended to be the first in a series of articles designed to help the beginning programmer make his/her programs more user-friendly. For the very beginner, this means making the programs so that someone who has never seen them and who knows nothing about computers can load it in and immediately be able to use it.

Wrap-around is when a computer is printing a word and reaches the edge of the screen (or page if you are using a printer) in the middle of it. Half of the word is on one line and half is on the next. Aside from various machine code routines (which I don't know anyway) such as the one in the MEMOTEXT unit that I am presently using, there are two common methods of handling the problem in BASIC. The first is to do the job manually. As you enter the data, using (for example):

```
10 INPUT A$
or
10 IF INKEY$="" THEN GOTO 10
20 LET A$=INKEY$
```

You can count the number of letters, spaces, etc. that you enter. You could also watch the screen. If you start printing 3 spaces from the left, then one line is used each time the print gets to the second space from the left. This is OK but can be very slow and awkward.

There is an easier way. Instead of you watching the input, have the computer adjust the output. Here is a routine reprinted from the ZX Printer User's booklet that shows how to do this. (Reprinted by permission of Sinclair Research, Ltd.)

```
10 REM TEXT EDIT
20 IF LEN A$>32 THEN GOTO 35
25 LPRINT A$
30 RETURN
35 LET C=0
40 IF C=32 THEN GOTO 65
45 LET B$=A$(32-C)
50 IF A$(33-C)=" " OR B$="," OR B$="," OR B$=":" OR
B$=";" OR B$="?" OR B$=CHR$ 11 THEN GOTO 70
55 LET C=C+1
60 GOTO 40
65 LET C=0
70 LPRINT A$( TO 32-C)
75 LET A$=A$(33-C TO )
80 IF A$(1)=" " THEN LET A$=A$(2 TO )
90 GOTO 20
```

Notice that the text to be edited is stored in A$. To use this to edit output to the screen instead of the printer, just change lines 25 and 70 to read PRINT instead of LPRINT. I use this as a routine to get instructions for games readable.

THE CONTINUING SAGA OF 64 COLUMNS

by TSUG STAFF, DAVE ALFORD, TIMEX STAFF, CAROLE CORCORAN JAMES HICKS AND MANY OTHERS

We still work on the 64 column mode. We managed to acquire a nice tape of "64-column Support" and have been using it and analysing it regularly. Below we have printed out a hex dump of the contents of the tape done by Dave Alford. He also did a disassembly of it, but we have not presented that disassembly here. Instead we have printed up a long document we got with the tape that was programmed by Carole Corcoran of TIMEX. In its present form, the tape consists of 920 bytes of machine code starting at address 57344; as is, the tape does nothing exciting. BUT we know it is fairly worthwhile and we think it is a series of patches or subroutines that can be called to write to the 2068 screen in 64-column mode, AND do other various and sundry things. The BIG PROBLEM is that the screen is darned hard to read with a fairly good 13" *monochrome* amber screen monitor while the computer is running in the 64-column mode. Jim Hicks says that using Dick Scoville's screen darkening routine overcomes these drawbacks. There is a bit of code missing from Carole Corcoran's commented disassembly, but many of you may enjoy getting your feet wet on this. We will be happy to supply a tape of the machine code for $5.00, if you would like.

Jim Hicks has been working on this tape and on this material for a good long while and has many of the routines working nicely, thus gaining great speed and flexibility over the cumbersome efforts that all of us have been attempting up until now in modified BASIC. Jim cannot be reached by phone unless you are a HAM, but his address is James Hicks, Rt. 1, Box 41-B, Bahama, NC 27503. Next month we will publish two more methods of attaching RGB monitors to the 2068; one method is *james hicks'* and the other is *Peter orvis'*. In the meantime, read John Dilger's explanation of what happens with his board and you will understand the circuits we present next month a bit better.

Dave Alford has also played around with this 64-column material some and his address is Dave Alford, 112 Koehler Crescent, Newport News, VA 23606. Dave has also worked on ROM calls in the Spectrum and 2068 along with John M. Brown, Jr. of Albuquerque. We hope to present that material next time in some useable fashion. Dave says: "The 21 18h at E0 2Eh and the 20h at E0 32h set the line length at 32 characters. Change them to correspond to the list for 64 columns by making them 41 18h and 40h respectively."

57344

E000H	20	C3	35	E0	00	00	C3	C1
E008H	E0	00	18	C3	7D	E1	38	C3
E010H	EF	E1	00	C3	06	E2	00	00
E018H	C3	0F	E2	C3	6B	E2	C3	72
E020H	E2	00	C3	94	E2	01	17	C3
E028H	D8	E0	17	01	00	3C	21	18
E030H	00	40	20	00	38	3A	00	E0
E038H	CD	90	E3	C5	ED	4B	2C	E0
E040H	EB	26	00	6F	29	29	29	09
E048H	C1	E6	07	3D	3A	32	E0	20
E050H	05	3C	05	4F	18	01	3C	B9
E058H	D5	CC	9B	E0	D1	C5	E5	3A
E060H	33	E0	06	FF	1F	38	01	04
E068H	1F	1F	9F	4F	3E	08	A7	EB
E070H	08	1A	A0	AE	A9	12	08	14
E078H	23	3D	20	F4	E1	C1	0D	79
E080H	CB	47	20	06	7C	F6	20	67
E088H	18	05	7C	E6	DF	67	23	CD
E090H	88	E3	0E	00	06	00	C9	0E
E098H	01	18	F9	3E	18	90	57	3A
E0A0H	2A	E0	BA	D2	5E	E3	3A	2B
E0A8H	E0	3D	20	0B	3E	01	32	2B
E0B0H	E0	C1	C1	0E	03	18	DD	32
E0B8H	2B	E0	ED	4B	25	E0	C3	E9
E0C0H	E0	ED	4B	04	E0	CD	4F	E3
E0C8H	3A	32	E0	3C	91	4F	3E	18
E0D0H	90	47	CD	5E	E3	C3	92	E0
E0D8H	ED	4B	25	E0	78	81	FE	19
E0E0H	D2	97	E0	CD	E9	E0	C3	92
E0E8H	E0	C5	3E	18	91	47	3A	32
E0F0H	E0	3C	4F	CD	77	E3	C1	C5
E0F8H	7D	A7	28	51	07	07	07	4F
E100H	3E	08	91	B8	38	3F	78	06
E108H	00	C5	47	0E	08	78	C5	0F
E110H	0F	0F	4F	06	00	EB	21	E0
E118H	FF	19	EB	C5	E5	ED	B0	E1
E120H	C1	7C	CB	6F	20	05	F6	20
E128H	67	18	EA	E6	DF	67	24	C1
E130H	0D	20	DA	C1	78	A7	28	04
E138H	2E	00	18	BC	C1	79	80	3D
E140H	4F	06	01	18	49	4F	78	91
E148H	47	C5	79	18	BD	05	C5	0E
E150H	08	C5	EB	21	E0	F8	19	EB
E158H	01	20	00	E5	ED	B0	E1	7C
E160H	CB	6F	20	05	F6	20	67	18
E168H	E9	E6	DF	67	24	C1	0D	20
E170H	E0	C1	78	A7	28	C6	11	20
E178H	F8	19	C3	F8	E0	ED	4B	09
E180H	E0	78	81	FE	19	D2	97	E0
E188H	CD	8E	E1	C3	92	E0	C5	3E
E190H	18	91	47	3A	32	E0	3C	4F
E198H	CD	77	E3	50	59	C1	D5	7D
E1A0H	07	07	07	4F	3E	08	91	B8
E1A8H	38	3D	78	06	00	C5	47	0E
E1B0H	08	78	C5	E6	07	0F	0F	0F
E1B8H	4F	06	00	0D	C5	E5	54	5D
E1C0H	36	00	13	ED	B0	E1	C1	7C
E1C8H	CB	6F	20	05	F6	20	67	18
E1D0H	EB	E6	DF	67	24	C1	0D	20
E1D8H	D8	C1	78	A7	28	05	2E	00
E1E0H	C3	9F	E1	C1	C3	5E	E3	4F
E1E8H	78	91	47	C5	79	18	BF	3A
E1F0H	0E	E0	E6	07	47	17	17	17
E1F8H	4F	F6	08	D3	FF	79	2F	B0
E200H	32	34	E0	C3	92	E0	3A	12
E208H	E0	32	33	E0	C3	92	E0	ED
E210H	4B	16	E0	CD	4F	E3	78	41
E218H	4F	2A	2C	E0	11	00	01	19
E220H	C5	78	1F	47	79	0F	0F	0F
E228H	E6	E0	A8	5F	79	E6	18	EE
E230H	40	57	C1	78	CB	47	28	04
E238H	7A	F6	20	57	06	60	C5	D5
E240H	E5	1A	AE	28	04	3C	20	17
E248H	3D	4F	06	07	14	23	1A	AE
E250H	A9	20	0C	10	F7	C1	C1	C1
E258H	3E	80	90	4F	06	00	C9	E1
E260H	11	00	00	19	D1	C1	10	D6
E268H	48	AF	C9	3A	34	E0	4F	06
E270H	00	C9	ED	4B	2E	E0	3E	18
E278H	90	47	3A	32	E0	3C	91	4F
E280H	FE	40	38	0B	0E	00	78	3C
E288H	47	FE	18	38	02	06	17	ED
E290H	43	04	E0	C9	3A	21	E0	A7
E298H	28	05	FE	06	C2	97	E0	47
E2A0H	3A	C2	5C	A7	20	25	B0	CA
E2A8H	92	E0	E5	D5	C5	21	C0	12
E2B0H	44	4D	11	40	08	19	ED	5B
E2B8H	65	5C	19	DA	0B	E3	ED	5B
E2C0H	B2	5C	A7	ED	52	D2	0B	E3
E2C8H	C1	D1	E1	CD	36	E3	78	F5
E2D0H	CD	8E	0E	CD	43	E3	F1	CD
E2D8H	10	E3	79	32	C2	5C	CB	77
E2E0H	28	15	3E	C1	B0	2F	CB	38
E2E8H	CB	38	CB	38	B0	32	34	E0
E2F0H	01	41	18	3E	40	18	05	3E
E2F8H	20	01	21	18	32	32	E0	21
E300H	00	00	22	7D	5C	CD	5E	E3
E308H	C3	92	E0	0E	02	C3	94	E0
E310H	4F	CB	47	20	16	FE	80	28
E318H	1A	CB	7F	C0	47	FE	02	C8
E320H	E6	C6	EE	06	C0	3E	40	B1
E328H	4F	AF	C9	FE	01	C0	06	01
E330H	0E	81	C9	AF	47	C9	F5	DB
E338H	FF	F6	80	D3	FF	3E	01	D3
E340H	F4	F1	C9	F5	AF	D3	F4	DB
E348H	FF	E6	3F	D3	FF	F1	C9	3E
E350H	17	B8	30	04	F1	C3	97	E0
E358H	3E	3F	B9	38	F7	C9	CD	77
E360H	E3	3A	32	E0	3C	91	CB	47
E368H	28	06	F5	3E	20	B4	67	F1
E370H	1F	5F	16	00	19	18	11	3E
E378H	18	90	57	0F	0F	0F	E6	E0
E380H	6F	7A	E6	18	F6	40	67	C9
E388H	ED	43	2E	E0	22	30	E0	C9
E390H	ED	4B	2E	E0	2A	30	E0	C9
E398H	00	00	00	00	00	00	00	00
E3A0H	00	00	00	00	00	00	00	00
E3A8H	00	00	00	00	00	00	00	00
E3B0H	00	00	00	00	00	00	00	00
E3B8H	00	00	00	00	00	00	00	00
E3C0H	00	00	00	00	00	00	00	00
E3C8H	00	00	00	00	00	00	00	00

```
                ;**********************************
                ;*                                *
                ;*             TIMEX              *
                ;* APPLICATION DEVELOPMENT LIBRARY *
                ;*                                *
                ;* NAME: 64-COL.MODE SUPPORT      *
                ;*                                *
                ;* ASC NO:  001                   *
                ;* VERSION: 001                   *
                ;* AUTHOR:  C. CORCORAN           *
                ;* DATE:    11/83                 *
                ;**********************************
#HEADING 64-COL.MODE SUPPORT
;
;
;                       ****DEFINITIONS****
;
        GLOBAL  WRCHAR,SETCUR,SETATT,SETMSK,CLRSCN,SCROLL,GTCHAR
        GLOBAL  GETATT,GETCUR
;
        GLOBAL  WRCHB,SETCUB,SETATB,SETMSB,CLRSCB,SCRLB,GTCHRB
        GLOBAL  GETATB,GETCUB
;
        GLOBAL  SETMODE,SETMDB
;
SCR_SZ          EQU     24      ; 24 LINES
CHRSET          EQU     3C00H   ; ROM CHAR.TABLE-100H
;
CHNG_VID        EQU     0E8EH   ; ROUTINE IN ROM EXTENSION
;
HREXPT          EQU     0FFH    ; HOME ROM EXT.SELECT PORT (BIT 7)
DKHSPT          EQU     0F4H    ; DOCK HORIZONTAL SELECT PORT
;
STKEND          EQU     5C65H   ; SYSTEM VARIABLES
UDG             EQU     5C7BH   ;
COORDS          EQU     5C7DH   ;
RAMTOP          EQU     5CB2H   ;
P_RAMT          EQU     5CB4H   ;
VIDMOD          EQU     5CC2H   ;
DRIVES          EQU     6840H   ;
INSERT_SZ       EQU     7B00H-DRIVES
MOVE_SZ         EQU     DRIVES-6000H
DEST7           EQU     OFFFFH-MOVE_SZ+1
FIX             EQU     DEST7-6000H
;
FIXTBL          EQU     1D00H
;
;
#HEADING 64-COL.MODE SUPPORT
;
;
;
E000            ORG     0E000H
;
;
;
;                       **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
;                                                               DECIMAL ADDRESS
;                                                               ---------------
E000 20         DATAB:  DEFB    20H     ; DATA BYTE FROM BASIC          57344
E001 C335E0     WRCHB:  JP      WRCHO   ; ENTRY TO WRITE CHARACTER      57345
E004 0000       LINCOL: DEFW    0       ; SET CURSOR PARAMETERS
                                        ;   LINCOL=COL. (0-63)          57348
                                        ;   LINCOL+1=LINE (0-23)        57349
E006 C3C1E0     SETCUB: JP      SETCBO  ; ENTRY TO SET CURSOR POSN.     57350
E009 0018       CLRCTL  DEFW    1800H   ; CLEAR SCREEN CONTROLS
                                        ;   CLRCTL=STARTING LINE NO.    57353
                                        ;   CLRCTL+1=LINE COUNT         57354
E00B C37DE1     CLRSCB: JP      CLRSBO  ; ENTRY TO CLEAR SCREEN         57355
E00E 38         ATTCTL  DEFB    38H     ; ATTRIBUTE CONTROL             57358
                                        ;   BIT 7=FLASH (0 IN 64-COL.)
                                        ;        6=BRIGHT(0 IN 64-COL.)
                                        ;        5
                                        ;        4=PAPER COLOR
                                        ;        3
                                        ;        2
                                        ;        1= INK COLOR
                                        ;        0
E00F C3EFE1     SETATB: JP      SETABO  ; ENTRY TO SET ATTRIBUTES       57359
E012 00         MSKCTL  DEFB    0       ; MASK CONTROL FROM BASIC:      57362
                                        ;   BIT 2=INVERSE
                                        ;     0=OVER
E013 C306E2     SETMSB: JP      SETMBO  ; ENTRY TO SET MASK CONTROLS    57363
E016 0000       GETCTL  DEFW    0       ; 'GET' CONTROL FROM BASIC      57366
                                        ; FORMAT IS AS FOR LINCOL
E018 C30FE2     GTCHRB: JP      GTCHBO  ; ENTRY TO GET CHARACTER        57368
E01B C368E2     GETATB: JP      GETABO  ; ENTRY TO GET ATTRIBUTE        57371
E01E C372E2     GETCUB: JP      GETCBO  ; ENTRY TO GET CURSOR POSN.     57374
E021 00         VIMODE  DEFB    0       ; VIDEO MODE CONTROL            57377
                                        ;       0=NORMAL
                                        ;       6=64-COL.MODE
E022 C394E2     SETMDB: JP      STMDBO  ; ENTRY TO SET VIDEO MODE       57378
E025 0117       SCRCTL  DEFW    1701H   ; SCROLL CONTROL
                                        ;   SCRCTL=STARTING LINE NO.    57381
                                        ;   SCRCTL+1=LINE COUNT         57382
E027 C3D8E0     SCRLB:  JP      SCRLBO  ; ENTRY TO SCROLL               57383
E02A 17         BOT_LN  DEFB    17H     ; BOTTOM LINE (LINE AFTER       57386
                                        ; WHICH SCREEN IS CONSIDERED
                                        ; FULL.  AUTOMATIC SCROLL
                                        ; WILL BE DONE IF SCRLCT
                                        ; DOES NOT DECREMENT TO 0.
E02B 01         SCRLCT  DEFB    1       ; SCROLL COUNT - 1 PLUS NO.     57387
                                        ; OF TIMES AUTOMATIC SCROLL
                                        ; WILL BE DONE.
;
;
;               **INTERNAL VARIABLES**
;
E02C 003C       CH_TBL  DEFW    CHRSET  ; ADDRESS OF CHARACTER TABLE (DEFAULTS
                                        ; TO TS2068 SYSTEM ROM)
E02E 4118       CURPOS  DEFW    1841H   ; CURRENT POSITION (INTERNAL FORMAT)
E030 0040       DF_ADRS DEFW    4000H   ; CURRENT DISPLAY FILE ADDRESS
E032 40         LIN_LEN DEFB    40H     ; LINE LENGTH (64)
E033 00         MASK_B  DEFB    0       ; MASK TO BE APPLIED TO DISPLAY CHARACTERS
E034 38         ATTBYT  DEFB    38H     ; CURRENT ATTRIBUTES
```

```
E063 06FF       LD      B,0FFH
E065 1F         RRA
E065 3801       JR      C,WRCH3         ; IF XORING NEW INTO OLD
E067 04         INC     B               ;   B=-1 IF XORING  =0 IF NOT
E068 1F         WRCH3   RRA
E069 1F         RRA
E06A 9F         SBC     A,A
E06B 4F         LD      C,A             ; C=-1 FOR INVERSE  =0 IF NOT
E06C 3E08       LD      A,8             ; SCAN COUNT
E06E A7         AND     A
E06F EB         WRCH4   EX      DE,HL
E070 08         WRCH5   EX      AF,AF'
E071 1A         LD      A,(DE)          ; OLD BIT ROW
E072 A0         AND     B
E073 AE         XOR     (HL)
E074 A9         XOR     C               ; INVERT IF C=-1
E075 12         LD      (DE),A          ; UPDATE DISPLAY FILE
E076 08         EX      AF,AF'
E077 14         INC     D               ; NEXT SCAN
E078 23         WRCH6   INC     HL      ; NEXT BYTE IN CHAR.FILE
E079 3D         DEC     A               ; TEST IF MORE SCANS
E07A 20F4       JR      NZ,WRCH5
E07C E1         WRCH7   POP     HL
E07D C1         POP     BC
E07E 0D         DEC     C               ; ADJUST CURSOR POSITION
E07F 79         LD      A,C             ; TEST IF EVEN/ODD
E080 CB47       BIT     0,A
E082 2006       JR      NZ,COLODD
E084 7C         COLEVN  LD      A,H
E085 F620       OR      20H
E087 67         LD      H,A
E088 1805       JR      WRCHEX
E08A 7C         COLODD  LD      A,H
E08B E6DF       AND     0DFH
E08D 67         LD      H,A
E08E 23         INC     HL              ; DF1 NEXT BYTE
E08F CD88E3     WRCHEX  CALL    STPOSN  ; STORE NEW POSITION
E092 0E00       GOODRET LD      C,0     ;   RETURN BC=0
E094 0600       ERRRET  LD      B,0     ; ENTER HERE WITH C NON-ZERO
E096 C9         RET
;
E097 0E01       INVPAR  LD      C,1     ; RETURN BC=1 FOR INVALID PARAMETERS
E099 18F9       JR      ERRRET
;
E09B 3E18       TVFULQ  LD      A,SCR_SZ        ; TEST IF NEW LINE IS OFF SCREEN
E09D 90         SUB     B               ;   A=LINE NO.
E09E 57         LD      D,A             ; SAVE IN D
E09F 3A2AE0     LD      A,(BOT_LN)      ; GET BOTTOM LINE
E0A2 BA         CP      D
E0A3 D25EE3     JP      NC,UPDPOSN      ; ON SCREEN - RETURN TO WRITE CHAR.
                                        ;  VIA UPDATE AND STORE POSITION
E0A6 3A2BE0     LD      A,(SCRLCT)      ; TEST SCROLL COUNT
E0A9 3D         DEC     A
E0AA 200B       JR      NZ,TVFUL1       ;   DO AUTOMATIC SCROLL USING SCRCTL
E0AC 3E01       LD      A,1             ; REINITIALIZE TO 1
E0AE 322BE0     LD      (SCRLCT),A
E0B1 C1         POP     BC
E0B2 C1         POP     BC              ; DISCARD RETURN TO WRITE CHAR.
E0B3 0E03       LD      C,3             ; RETURN BC=3 FOR SCREEN FULL
E0B5 18DD       JR      ERRRET
E0B7 322BE0     TVFUL1  LD      (SCRLCT),A      ; UPDATE VARIABLE
E0BA ED4B25E0   LD      BC,(SCRCTL)     ; GET SCROLL CONTROLS (STARTING LINE
                                        ;   AND LINE COUNT)
E0BE C3E9E0     JP      SCRLO           ; RETURN TO WRITE CHAR. VIA SCROLL
;
#HEADING 64-COL.SET CURSOR
;
;
SETCBO:                                 ; HERE FROM BASIC ENTRY.  PARAMETERS
                                        ; IN LINCOL
E0C1 ED4B04E0   LD      BC,(LINCOL)
;
SETCUR:                                 ; ENTRY WITH LINE/COL. IN BC REG.
E0C5 CD4FE3     CALL    TSTPAR          ; TEST PARAMETERS FOR VALIDITY
E0C8 3A32E0     LD      A,(LIN_LEN)
E0CB 3C         INC     A
E0CC 91         SUB     C
E0CD 4F         LD      C,A             ; CONVERT TO INTERNAL FORMAT
E0CE 3E18       LD      A,SCR_SZ
E0D0 90         SUB     B
E0D1 47         LD      B,A
E0D2 CD5EE3     CALL    UPDPOSN         ; CALCULATE AND STORE POSITION
E0D5 C392E0     JP      GOODRET         ; RETURN BC=0
;
#HEADING 64-COL.SCROLL
;
;
;
SCRLBO:                                 ; HERE FROM BASIC ENTRY TO SCROLL
                                        ; SCREEN
E0D8 ED4B25E0   LD      BC,(SCRCTL)     ; GET CONTROL INFO.
;
SCROLL:                                 ; HERE WITH CONTROLS IN BC
                                        ;  B=NO. OF LINES
                                        ;  C=STARTING LINE NO.
E0DC 78         LD      A,B             ; TEST VALIDITY
E0DD 81         ADD     A,C
E0DE FE19       CP      25
E0E0 D297E0     JP      NC,INVPAR       ; ERROR IF B+C>24
E0E3 CDE9E0     CALL    SCRLO           ; DO SCROLL
E0E6 C392E0     JP      GOODRET         ; RETURN BC=0
;
E0E9 C5         SCRLO   PUSH    BC
E0EA 3E18       LD      A,SCR_SZ        ; GET STARTING LINE IN INTERNAL FORMAT
E0EC 91         SUB     C
E0ED 47         LD      B,A
E0EE 3A32E0     LD      A,(LIN_LEN)
E0F1 3C         INC     A               ; COL. 0
E0F2 4F         LD      C,A
E0F3 CD77E3     CALL    LN_BU
E0F6 C1         POP     BC
E0F7 C5         PUSH    BC              ; SAVE ORIG.BC FOR EXIT
E0F8 7D         TESTAD  LD      A,L     ; TEST IF AT START OF BLOCK
E0F9 A7         AND     A
E0FA 2851       JR      Z,STBLK         ; YES
E0FC 07         RLCA
E0FD 07         RLCA
```

```
E0FE  07          285          RLCA
E0FF  4F          286          LD    C,A              ! GET NO. OF LINES THIS BLOCK
E100  3E08        287          LD    A,8
E102  91          288          SUB   C
E103  B8          289          CP    B                ! TEST AGAINST TOTAL LINES
E104  383F        290          JR    C,GRBLK          ! TOTAL GREATER THAN THIS BLOCK
E106  78          291  INSDBLK      LD    A,B
E107  0600        292          LD    B,0              ! REMAINING COUNT
E109  C5          293          PUSH  BC
E10A  47          294  LOOP    LD    B,A              ! B=LINES THIS BLOCK
E10B  0E08        295          LD    C,8              ! C=SCAN COUNT
E10D  78          296  LOOPO   LD    A,B
E10E  C5          297          PUSH  BC               ! SAVE SCAN COUNT
E10F  0F          298          RRCA
E110  0F          299          RRCA
E111  0F          300          RRCA                   ! CALCULATE NO. OF BYTES
E112  4F          301          LD    C,A
E113  0600        302          LD    B,0              ! BC=32*# OF LINES
E115  EB          303  LOOP1   EX    DE,HL
E116  21E0FF      304          LD    HL,-20H          ! GET ADRS. OF PREV. LINE
E119  19          305          ADD   HL,DE
E11A  EB          306          EX    DE,HL            ! TO DE
E11B  C5          307          PUSH  BC               ! SAVE COUNT
E11C  E5          308          PUSH  HL               ! SAVE ADDRESS
E11D  EDB0        309          LDIR                   ! DO MOVE
E11F  E1          310          POP   HL
E120  C1          311          POP   BC
E121  7C          312          LD    A,H
E122  CB6F        313          BIT   5,A              ! TEST IF DF1
E124  2005        314          JR    NZ,NXTSC         ! NO
E126  F620        315          OR    20H
E128  67          316          LD    H,A              ! DO DF2
E129  18EA        317          JR    LOOP1
E12B  E6DF        318  NXTSC   AND   0DFH            ! BACK TO DF1
E12D  67          319          LD    H,A
E12E  24          320          INC   H                ! NEXT SCAN LINE
E12F  C1          321          POP   BC               ! SCAN COUNT
E130  0D          322          DEC   C
E131  20DA        323          JR    NZ,LOOPO         ! MORE SCANS
E133  C1          324          POP   BC               ! RESIDUAL LINE COUNT
E134  78          325          LD    A,B
E135  A7          326          AND   A
E136  2804        327          JR    Z,SCRLXT         ! DONE
E138  2E00        328          LD    L,0              ! START OF NEXT BLOCK
E13A  18BC        329          JR    TESTAD           ! DO REMAINING LINE(S)
E13C  C1          330  SCRLXT  POP   BC               ! ORIG.BC
E13D  79          331          LD    A,C              ! STARTING LINE
E13E  80          332          ADD   A,B              ! ADD NO. OF LINES MOVED
E13F  3D          333          DEC   A                ! LAST LINE SCROLLED
E140  4F          334          LD    C,A
E141  0601        335          LD    B,1              ! CLEAR 1 LINE
E143  1849        336          JR    CLRSCO           ! CLEAR VACATED LINE AND RETURN
                  337  !
E145  4F          338  GRBLK   LD    C,A
E146  78          339          LD    A,B
E147  91          340          SUB   C                ! ADJUST TOTAL LINE COUNT BY NO.
E148  47          341          LD    B,A              ! OF LINES THIS BLOCK
E149  C5          342          PUSH  BC
E14A  79          343          LD    A,C              ! LOAD A NO. OF LINES THIS BLOCK
E14B  18BD        344          JR    LOOP
                  345  !
E14D  05          346  STBLK   DEC   B                  ! ADJUST TOTAL LINE COUNT-1
E14E  C5          347          PUSH  BC
E14F  0E08        348          LD    C,8              ! SCAN COUNT
E151  C5          349  STBLKO  PUSH  BC               ! SAVE SCAN COUNT
E152  EB          350  STBLK1  EX    DE,HL
E153  21E0F8      351          LD    HL,-720H
E156  19          352          ADD   HL,DE
E157  EB          353          EX    DE,HL            ! ADRS. OF PREV. LINE
E158  012000      354          LD    BC,32            ! BYTE COUNT
E15B  E5          355          PUSH  HL
E15C  EDB0        356          LDIR                   ! DO MOVE
E15E  E1          357          POP   HL
E15F  7C          358          LD    A,H
E160  CB6F        359          BIT   5,A
E162  2005        360          JR    NZ,STBLK2
E164  F620        361          OR    20H              ! DO DF2
E166  67          362          LD    H,A
E167  18E9        363          JR    STBLK1
E169  E6DF        364  STBLK2  AND   0DFH             ! BACK TO DF1
E16B  67          365          LD    H,A
E16C  24          366          INC   H                ! TO NEXT SCAN LINE
E16D  C1          367          POP   BC
E16E  0D          368          DEC   C                ! TEST IF MORE SCANS
E16F  20E0        369          JR    NZ,STBLKO
E171  C1          370          POP   BC               ! REMAINING LINE COUNT
E172  78          371          LD    A,B
E173  A7          372          AND   A
E174  28C6        373          JR    Z,SCRLXT
E176  1120F8      374          LD    DE,-7E0H         ! ADJUST TO LINE 1
E179  19          375          ADD   HL,DE
E17A  C3F8E0      376          JP    TESTAD           ! BC=LINE COUNT
                  377                                 ! HL=DF ADDRESS
                  378  !

                  379  *HEADING 64-COL.CLEAR SCREEN
                  380  !
                  381  !
                  382  CLRSBO:                        ! HERE FROM BASIC ENTRY TO CLEAR SCREEN
E17D  ED4B09E0    383          LD    BC,(CLRCTL)      ! GET CONTROL INFO.
                  384  !
                  385  CLRSCN:                        ! ENTRY WITH BC=LINE COUNT AND
                  386                                 ! STARTING LINE NO. FOR CLEAR
E181  78          387          LD    A,B
E182  81          388          ADD   A,C
E183  FE19        389          CP    25
E185  D297E0      390          JP    NC,INVPAR        ! ERROR IF B+C>24
E188  CD8EE1      391          CALL  CLRSCO           ! DO SERVICE
E18B  C392E0      392          JP    GOODRET          ! RETURN BC=0
                  393  !
                  394  !
E18E  C5          395  CLRSCO  PUSH  BC
E18F  3E18        396          LD    A,SCR_SZ         ! CONVERT TO INTERNAL FORMAT
E191  91          397          SUB   C
E192  47          398          LD    B,A
E193  3A32E0      399          LD    A,(LIN_LEN)
E196  3C          400          INC   A                ! SET TO COL.0
E197  4F          401          LD    C,A
```

```
E198  CD77E3      402          CALL  LN_BU            ! GET ADDRESS
E19B  50          403          LD    D,B
E19C  59          404          LD    E,C              ! SAVE POSITION
E19D  C1          405          POP   BC               ! ORIG. BC
E19E  D5          406          PUSH  DE
E19F  7D          407  CLRSC1  LD    A,L              ! GET # OF LINES THIS BLOCK
E1A0  07          408          RLCA
E1A1  07          409          RLCA
E1A2  07          410          RLCA
E1A3  4F          411          LD    C,A
E1A4  3E08        412          LD    A,8
E1A6  91          413          SUB   C
E1A7  B8          414          CP    B                ! COMPARE TO TOTAL LINE COUNT
E1A8  383D        415          JR    C,CLRSC7
E1AA  78          416  CLRSC2  LD    A,B              ! NO. OF LINES
E1AB  0600        417          LD    B,0              ! REMAINING LINES
E1AD  C5          418          PUSH  BC
E1AE  47          419  CLRSC3  LD    B,A
E1AF  0E08        420          LD    C,8              ! SCAN COUNT
E1B1  78          421  CLRSC4  LD    A,B
E1B2  C5          422          PUSH  BC
E1B3  E607        423          AND   7
E1B5  0F          424          RRCA
E1B6  0F          425          RRCA
E1B7  0F          426          RRCA
E1B8  4F          427          LD    C,A              ! BC=32*NO. OF LINES
E1B9  0600        428          LD    B,0              ! (=0 IF 256 FOR 8 LINES)
E1BB  0D          429          DEC   C                ! ADJUST
E1BC  C5          430  CLRSC5  PUSH  BC
E1BD  E5          431          PUSH  HL
E1BE  54          432          LD    D,H
E1BF  5D          433          LD    E,L
E1C0  3600        434          LD    (HL),00          ! CLEAR DF
E1C2  13          435          INC   DE
E1C3  EDB0        436          LDIR
E1C5  E1          437          POP   HL
E1C6  C1          438          POP   BC
E1C7  7C          439          LD    A,H
E1C8  CB6F        440          BIT   5,A
E1CA  2005        441          JR    NZ,CLRSC6
E1CC  F620        442          OR    20H
E1CE  67          443          LD    H,A              ! DO DF2
E1CF  18EB        444          JR    CLRSC5
E1D1  E6DF        445  CLRSC6  AND   0DFH             ! RESTORE DF1
E1D3  67          446          LD    H,A
E1D4  24          447          INC   H                ! NEXT SCAN ROW
E1D5  C1          448          POP   BC               ! POP SCAN COUNT
E1D6  0D          449          DEC   C
E1D7  20D8        450          JR    NZ,CLRSC4
E1D9  C1          451          POP   BC               ! TOTAL LINE COUNT
E1DA  78          452          LD    A,B
E1DB  A7          453          AND   A
E1DC  2805        454          JR    Z,CLRSXT
E1DE  2E00        455          LD    L,0              ! HL=START OF NEXT BLOCK
E1E0  C39FE1      456          JP    CLRSC1           ! B=REMAINING LINE COUNT
E1E3  C1          457  CLRSXT  POP   BC               ! POSITION
E1E4  C35EE3      458          JP    UPDPOSN          ! RETURN VIA UPDATE AND STORE POSITION
E1E7  4F          459  CLRSC7  LD    C,A              ! SAVE NO. LINES THIS BLOCK
E1E8  78          460          LD    A,B
E1E9  91          461          SUB   C                ! ADJUST TOTAL LINE COUNT
E1EA  47          462          LD    B,A
E1EB  C5          463          PUSH  BC               ! REMAINING LINE COUNT
E1EC  79          464          LD    A,C              ! LINES THIS BLOCK
E1ED  18BF        465          JR    CLRSC3           !
                                                      MSM 5.7

                  466  *HEADING 64-COL.MODE ATTRIBUTES
                  467  !
                  468  !
                  469  !
                  470  SETABO:                        ! HERE FROM BASIC ENTRY TO SET ATTRIBUTI
E1EF  3A0EE0      471          LD    A,(ATTCTL)       ! GET CONTROL INFO.
                  472  !
                  473  SETATT:                        ! ENTRY WITH ATTRIBUTE BYTE IN A
E1F2  E607        474          AND   7
E1F4  47          475          LD    B,A              ! SAVE INK SELECTION
E1F5  17          476          RLA
E1F6  17          477          RLA
E1F7  17          478          RLA                    ! SHIFT TO H/W POSITION
E1F8  4F          479          LD    C,A
E1F9  F606        480          OR    6                ! SET MODE
E1FB  D3FF        481          OUT   (HREXPT),A       ! SET H/W
E1FD  79          482          LD    A,C              ! BITS IN 3-5
E1FE  2F          483          CPL                    ! GET COMPLEMENTARY COLOR
E1FF  B0          484          OR    B                ! COMBINE WITH INK
E200  3234E0      485          LD    (ATTBYT),A       ! SAVE IN VARIABLE
E203  C392E0      486          JP    GOODRET          ! RETURN BC=0
                  487  !
                  488  !
                  489  !
                  490  SETMBO:                        ! HERE FROM BASIC ENTRY TO SET MASK
E206  3A12E0      491          LD    A,(MSKCTL)       ! GET CONTROL INFO.
                  492  !
                  493  SETMSK:                        ! ENTRY WITH MASK VALUE IN A
E209  3233E0      494          LD    (MASK_B),A       ! STORE MASK FOR APPLICATION TO FUTURE
E20C  C392E0      495          JP    GOODRET          ! DISPLAYS
                  496  !
                  497  *HEADING 64-COL.'GET' ROUTINES
                  498  !
                  499  !
                  500  GTCHBO:                        ! HERE FROM BASIC ENTRY TO GET CHARACTI
                  501                                 ! (RETURNS CODE FOR CHARACTER AT
                  502                                 ! POSITION SPECIFIED IN GETCTL)
E20F  ED4B16E0    503          LD    BC,(GETCTL)      ! GET CONTROL INFO.
                  504  !
                  505  GTCHAR:                        ! ENTRY WITH POSITION IN BC
E213  CD4FE3      506          CALL  TSTPAR           ! TEST PARAMETERS
E216  78          507          LD    A,B
E217  41          508          LD    B,C
E218  4F          509          LD    C,A              ! B=COL.  C=LINE FOR THIS RTN.
E219  2A2CE0      510          LD    HL,(CH_TBL)      ! CHAR.TABLE
E21C  110001      511          LD    DE,8*20H
E21F  19          512          ADD   HL,DE            ! HL=1ST CHAR.IN TABLE
E220  C5          513          PUSH  BC
E221  78          514          LD    A,B
E222  1F          515          RRA                    ! COL/2=POSN IN DISPLAY FILE
```

```
E223  47        516             LD    B,A
E224  79        517             LD    A,C
E225  0F        518             RRCA
E226  0F        519             RRCA
E227  0F        520             RRCA
E228  E6E0      521             AND   0E0H
E22A  A8        522             XOR   B
E22B  5F        523             LD    E,A
E22C  79        524             LD    A,C
E22D  E618      525             AND   18H
E22F  EE40      526             XOR   40H
E231  57        527             LD    D,A                 ; DE=CHAR. IN DISPLAY FILE
E232  C1        528             POP   BC
E233  78        529             LD    A,B
E234  CB47      530             BIT   0,A
E236  2804      531             JR    Z,GTCH1
E238  7A        532             LD    A,D
E239  F620      533             OR    20H                 ; IN DF2
E23B  57        534             LD    D,A
E23C  0660      535 GTCH1       LD    B,96                ; NO. OF PRINTABLE CHARACTERS
E23E  C5        536 GTCH2       PUSH  BC                  ; SAVE COUNT
E23F  D5        537             PUSH  DE                  ; SAVE ADRS. IN DF
E240  E5        538             PUSH  HL                  ; SAVE ADRS. IN CHAR.TABLE
E241  1A        539             LD    A,(DE)
E242  AE        540             XOR   (HL)                ; TEST AGAINST CHAR. SET
E243  2804      541             JR    Z,GTCH3             ; MATCH
E245  3C        542             INC   A                   ; TEST INVERSE
E246  2017      543             JR    NZ,GTCH7
E248  3D        544             DEC   A                   ; A=-1 FOR INVERSE
E249  4F        545 GTCH3       LD    C,A                 ; C=0 FOR MATCH  -1 FOR INVERSE
E24A  0607      546             LD    B,7
E24C  14        547 GTCH5       INC   D                   ; NEXT SCAN
E24D  23        548             INC   HL
E24E  1A        549             LD    A,(DE)
E24F  AE        550             XOR   (HL)
E250  A9        551             XOR   C
E251  200C      552             JR    NZ,GTCH7            ; NO MATCH
E253  10F7      553             DJNZ  GTCH5       ; FOR NEXT SCAN
                554
                555                                       ; HERE IF MATCH WITH (SP)=CHAR IN CHAR SET
                556                                       ; (SP+2)=CHAR.IN DISPLAY FILE (SP+5)=NO.
                557                                       ; COUNT
E255  C1        558             POP   BC
E256  C1        559             POP   BC
E257  C1        560             POP   BC
E258  3E80      561             LD    A,80H
E25A  90        562             SUB   B                   ; A=CHAR.CODE
E25B  4F        563             LD    C,A
E25C  0600      564             LD    B,0
E25E  C9        565             RET                       ; BC=CHAR.CODE
                566 ;
E25F  E1        567 GTCH7       POP   HL                  ; HERE WHEN NO MATCH
E260  110800    568             LD    DE,8                ; LOOK AT NEXT CHAR. IN SET
E263  19        569             ADD   HL,DE
E264  D1        570             POP   DE                  ; CHAR. IN DF
E265  C1        571             POP   BC                  ; CHAR.COUNT
E266  10D6      572             DJNZ  GTCH2               ; LOOK AGAIN
                573                                       ;
                574                                       ; HERE IF NO MATCH AT ALL
E268  48        575             LD    C,B
E269  AF        576             XOR   A                   ; SET BC AND A=ZERO
E26A  C9        577             RET
                578
                579                                       ; GET ATTRIBUTE BYTE
                580                                       ; NOTE THAT IN 64-COL.MODE ALL SCREEN POSITIONS
                581                                       ; HAVE THE SAME ATTRIBUTES. THEREFORE IT IS
                582                                       ; NOT NECESSARY TO USE THE 'GETCTL' POSITION
                583                                       ; PARAMETERS
                584 ;
                585 GETABO:                               ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                586 ;
E26B  3A34E0    587 GETATT:     LD    A,(ATTBYT)          ; GET ATTRIBUTE BYTE
E26E  4F        588             LD    C,A                 ; PUT IN BC
E26F  0600      589             LD    B,0
E271  C9        590             RET
                591 ;                                     ; GET CURSOR POSITION
                592 GETCBO:                               ; HERE FROM BASIC ENTRY
                593 GETCUR:
E272  ED4B2EE0  594             LD    BC,(CURPOS)         ; GET INTERNAL POSITION
E276  3E18      595             LD    A,SCR_SZ            ; CONVERT TO USER FORMAT
E278  90        596             SUB   B
E279  47        597             LD    B,A                 ; B=LINE NO. (0-23)
E27A  3A32E0    598             LD    A,(LIN_LEN)
E27D  3C        599             INC   A
E27E  91        600             SUB   C
E27F  4F        601             LD    C,A                 ; C=COL.NO.(0-63)
E280  FE40      602             CP    64                  ; TEST IF END OF LINE
E282  3808      603             JR    C,GETC1             ; NO
E284  0E00      604             LD    C,0                 ; NEXT POSN.START OF NEXT LINE
E286  78        605             LD    A,B
E287  3C        606             INC   A                   ; BUMP TO NEXT LINE
E288  47        607             LD    B,A
E289  FE18      608             CP    24                  ; TEST IF OFF SCREEN
E28B  3802      609             JR    C,GETC1             ; NO
E28D  0617      610             LD    B,23                ; POSN. IS AT BOTTOM LINE
E28F  ED4304E0  611 GETC1       LD    (LINCOL),BC         ; STORE FOR BASIC PROGRAM
E293  C9        612             RET                       ; VALUES ALSO IN BC
                613 ;
                614 *HEADING 64-COL.CHANGE MODE
                615 ;
                616 ;
                617 STMDBO:                               ; HERE FROM BASIC ENTRY WITH MODE IN
                618                                       ; 'VIMODE'
E294  3A21E0    619             LD  A,(VIMODE)
                620 ;
                621 SETMODE:                              ; ENTRY WITH MODE IN A
                622 ;
E297  A7        623             AND   A
E298  2805      624             JR    Z,SETMD1            ; TEST FOR VALIDITY
```

```
E29A  FE06      625             CP    6
E29C  C297E0    626             JP    NZ,INVPAR           ; INVALID PARAMETERS
                627 ;
E29F  47        628 SETMD1      LD    B,A                 ; SAVE MODE
E2A0  3AC25C    629             LD    A,(VIDMOD)
E2A3  A7        630             AND   A                   ; TEST CURRENT MODE
E2A4  2025      631             JR    NZ,SETMD2
E2A6  B0        632             OR    B                   ; CURRENT MODE=0  TEST IF NEW MODE=0
E2A7  CA92E0    633             JP    Z,GOODRET           ; RETURN BC=0
E2AA  E5        634             PUSH  HL                  ; SAVE REGS. WHILE TEST ROOM
E2AB  D5        635             PUSH  DE
E2AC  C5        636             PUSH  BC
E2AD  21C012    637             LD    HL,INSERT_SZ
E2B0  44        638             LD    B,H
E2B1  4D        639             LD    C,L
E2B2  114008    640             LD    DE,MOVE_SZ
E2B5  19        641             ADD   HL,DE               ; HL=TOTAL ROOM NEEDED
E2B6  ED5B655C  642             LD    DE,(STKEND)
E2BA  19        643             ADD   HL,DE               ; ADD MEMORY IN USE
E2BB  DA0BE3    644             JP    C,EXT_NRM           ; NOT ENOUGH MEMORY TO OPEN DF2
E2BE  ED5B825C  645             LD    DE,(RAMTOP)
E2C2  A7        646             AND   A
E2C3  ED52      647             SBC   HL,DE
E2C5  D20BE3    648             JP    NC,EXT_NRM          ; NOT ENOUGH MEMORY
E2C8  C1        649             POP   BC                  ; RESTORE REGS.
E2C9  D1        650             POP   DE
E2CA  E1        651             POP   HL
E2CB  CD36E3    652 SETMD2      CALL  ENBLEXT             ; ENABLE ROM EXT.
E2CE  78        653             LD    A,B                 ; MODE TO A
E2CF  F5        654             PUSH  AF                  ; SAVE MODE
E2D0  CD3E0E    655             CALL  CHNG_VID            ; CALL RTN. TO OPEN/CLOSE 2ND DF
E2D3  CD43E3    656             CALL  ENBLHOME            ; REENABLE HOME BANK
E2D6  F1        657             POP   AF
E2D7  CD10E3    658             CALL  GETVAL              ; H/W MODE TO B; VIDMOD VALUE TO C
E2DA  79        659             LD    A,C
E2DB  32C25C    660             LD    (VIDMOD),A; UPDATE VIDMOD
E2DE  CB77      661 UPDATE      BIT   6,A                 ; UPDATE SYSTEM VARS. BASED ON MODE
E2E0  2815      662             JR    Z,UPDT1
E2E2  3EC1      663 SXT4        LD    A,0C1H              ; 64-COL. MODE
E2E4  B0        664             OR    B                   ; SIGHT ATTRIBUTE BYTE BASED
E2E5  2F        665             CPL                       ; ON INK SELECTION
E2E6  CB39      666             SRL   B
E2E8  CB39      667             SRL   B
E2EA  CB39      668             SRL   B
E2EC  B0        669             OR    B
E2ED  3279E0    670             LD    (ATTBYT),A          ; ATTRIBUTE BYTE
E2F0  011810    671             LD    BC,1018H            ; COL.0 LINE 0
E2F3  3E40      672             LD    A,40H               ; LINE LENGTH
E2F5  1805      673             JR    UPDT2
E2F7  3E20      674 UPDT1       LD    A,20H               ; LINE LENGTH 32
E2F9  012118    675             LD    BC,1821H            ; COL.0 LINE 0
E2FC  3232E0    676 UPDT2       LD    (LIN_LEN),A
E2FF  210000    677             LD    HL,0
E302  227D5C    678             LD    (COORDS),HL         ; INITIALIZE PLOT POSITION
E305  CD5EE3    679             CALL  UPDPOSN             ; UPDATE AND STORE HOME POSITION
E308  C392E0    680             JP    GOODRET             ; RETURN BC=0
E30B  0E02      681 EXT_NRM     LD    C,2                 ; RETURN BC=2 FOR NO ROOM
E30D  C394E0    682             JP    ERRRET
                683 ;
                684 *HEADING  64-COL.MODE SUPPORT SUB-RTNS.
                685 ;
                686 ;
                687 ;
                688                                       ; INPUT:  VIDEO MODE IN A
                689                                       ; OUTPUT: H/W SETTING IN B
                690                                       ;         VIDMOD SETTING IN C
                691                                       ; RETURNS NZ STATUS IF MODE INVALID
                692                                       ;
                693                                       ;  VALID VALUES:  INPUT     H/W     VIDMOD
                694                                       ;                  -----     ---     ------
                695                                       ;   *=DISPLAY FILE ACTIVE AT SCREEN
                696                                       ;
                697                                       ;    DF1 ONLY       0        0        0
                698                                       ;   DF1*& DF2       80       0        80
                699                                       ;   DF1 & DF2*      1        1        81
                700                                       ;   DATA TO DF1     4        1        84
                701                                       ;       (DF2*)
                702                                       ;   DATA TO DF2     5        0        85
                703                                       ;       (DF1*)
                704                                       ;  HIGH RES.OR.    2        2        2
                705                                       ;  64-COLUMN       06 - 3E  06 - 3E  46 - 7E
E310  4F        706 GETVAL      LD    C,A
E311  CB47      707             BIT   0,A
E313  2016      708             JR    NZ,TST01
E315  FE80      709             CP    80H
E317  281A      710             JR    Z,SET0
E319  CB7F      711             BIT   7,A
E31B  C0        712             RET   NZ
E31C  47        713             LD    B,A
E31D  FE02      714             CP    2
E31F  C8        715             RET   Z
E320  E6C6      716             AND   0C6H
E322  EE06      717             XOR   6
E324  C0        718             RET   NZ
E325  3E40      719             LD    A,40H
E327  B1        720             OR    C
E328  4F        721             LD    C,A
E329  AF        722             XOR   A
E32A  C9        723             RET
E32B  FE01      724 TST01       CP    1
E32D  C0        725             RET   NZ
E32E  0601      726             LD    B,1
E330  0E81      727             LD    C,81H
E332  C9        728             RET
E333  AF        729 SET0        XOR   A
E334  47        730             LD    B,A
E335  C9        731             RET
                732 ;
                733 ;
                734                                       ; ROUTINES TO ENABLE CHRTR 0 IN ROM EXT
                735                                       ; FOR ACCESS TO CHNG_VID ROUTINE
E336  F5        736 ENBLEXT       PUSH  AF
E337  DBFF      737               IN    A,(HREXPT)
E339  F680      738               OR    80H               ; SELECT ROM EXT.
E33B  D3FF      739               OUT   (HREXPT),A
E33D  3E01      740               LD    A,1
E33F  D3F4      741               OUT   (DKHSPT),A
```

```
E341  F1          742          POP   AF
E342  C9          743          RET
                  744  !
E343  F5          745  ENBLHOME  PUSH  AF
E344  AF          746          XOR   A
E345  D3F4        747          OUT   (DISHSPT),A
E347  DBFF        748          IN    A,(HREXPT)
E349  E63F        749          AND   03FH
E34B  D3FF        750          OUT   (HREXPT),A
E34D  F1          751          POP   AF
E34E  C9          752          RET
                  753  !
                  754                          ! TEST PARAMETER VALIDITY
                  755                          ! IF NOT VALID, DISCARDS RETURN
                  756                          ! AND EXITS VIA INVPAR
E34F  3E17        757  TSTPAR   LD    A,23      ! TEST LINE >23
E351  B8          758          CP    B
E352  3004        759          JR    NC,TSTPR1
E354  F1          760  PARERR   POP   AF
E355  C397E0      761          JP    INVPAR
E358  3E3F        762  TSTPR1   LD    A,63
E35A  B9          763          CP    C         ! TEST COL. >63
E35B  38F7        764          JR    C,PARERR
E35D  C9          765          RET
                  766  !
                  767  !
                  768  !
                  769  UPDPOSN                  ! ENTER HERE WITH BC=LINE/COL.IN
                  770                          ! INTERNAL FORMAT
E35E  CD77E3      771          CALL  LN_BU       ! GET DISPLAY FILE ADRS.
E361  3A32E0      772          LD    A,(LIN_LEN) ! TEST WHICH DF
E364  3C          773          INC   A
E365  91          774          SUB   C
E366  CB47        775          BIT   0,A
E368  2806        776          JR    Z,UPDPO1
E36A  F5          777          PUSH  AF
E36B  3E20        778          LD    A,20H
E36D  B4          779          OR    H           ! IN DF2
E36E  67          780          LD    H,A
E36F  F1          781          POP   AF
E370  1F          782          RRA               ! COL/2 IS POSITION IN DISPLAY FILE
E371  [illegible] 783          [illegible]
E372  1600        784          LD    D,0
```

Note MISSING LINES

```
64-COL.MODE SUPPORT SUB-RTNS  IO_64
 LOC   OBJ CODE M STMT SOURCE STATEMENT
E382  E618        800          AND   18H
E384  F640        801          OR    40H
E386  67          802          LD    H,A
E387  C9          803          RET
                  804
                  805  STPOSN:                  ! STORE CURSOR POSITION
E388  ED432EE0    806          LD    (CURPOS),BC
E38C  2230E0      807          LD    (DF_ADRS),HL
E38F  C9          808          RET
                  809  !
                  810  LDPOSN:                  ! LOAD CURSOR POSITION
E390  ED4B2EE0    811          LD    BC,(CURPOS)
E394  2A30E0      812          LD    HL,(DF_ADRS)
E397  C9          813          RET
                  814  !
                  815          END
```

FALL OF THE THIRD REICH
by MARK STUEBER
Reviewed by ROBERT CROMARTIE

I played Mark Stuber's Fall of the Third Reich and found it to be essentially identical to his first effort. War in the East, which I wrote about in these pages not long ago. In this game, you are (as in the first) in charge of the Nazi forces, this time attempting to defend Berlin against the onslaught of the Allies and the Russians in the waning days of WWII. If taking Moscow within the time limit of War in the East was hard, holding Berlin in Fall of the Third Reich is as close to impossible as can be imagined (in the instructions the author himself concedes it is "a nightmare for the player").

The mechanics of the game are as in War in the East, with the difference that during the attack phase you can choose whether or not to attack with a given unit, thus allowing you to avoid "kamikaze" missions by weak units against stong defenders. Despite this improvement, the basic weaknesses of the first game (as detailed in my first review) remain, and my overall impression is the same.

FOR $17.95

ORDER FROM

+$2.00 S+H

SHARP'S

127 NINE MILE RD. • SANDSTON, VA 23150 • (804) 737-4895

OLIGER 2068 EXPANSION RGB CIRCUIT INSTRUCTIONS

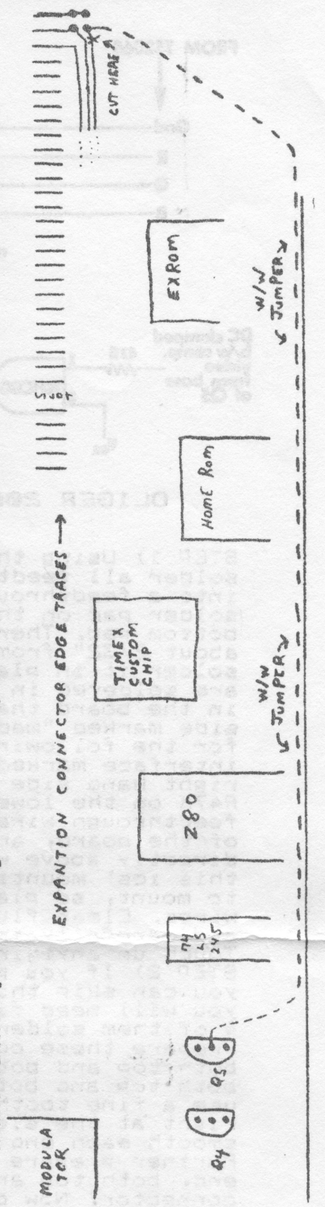
Although this RGB circuit should work with any RGB monitor, these instructions will detail its' connection to the Amdek Color II+ monitor only because this is the particular monitor I chose to purchase. This does not mean that this monitor is the best value around, but just that it is a very popular model and the one I have. If your RGB monitor can be configured to the Apple II compatable, via the use of a switch or other means, then connection to your monitor will be very similar if not identical. If your monitor can only be IBM compatible, then you will need to use the positive sync outputs and to connect its' intensity input high (to +5v) rather than gnd to get the correct color yellow.

All the outputs used to connect to your monitor are labeled on the board per the schematic below. Wires are soldered to these "doughnuts", with the wire extending through the board and soldered both on the boards' top & bottom. The wires or cable should be brought off the board to the rear, right over the part of the board labeled "clamp". Put a piece of insulation around individual wires at this point if a cable is not used, and use a small piece of solid bare 22AWG wire to form a cable clamp by bending it into a U and inserting it over the cable and into the clamp holes. Solder in place on the boards' bottom. The insulation over individual wires will keep the heat of soldering from damaging the board & the clamp itself keeps strain from board movement from breaking off any of the wires from the board in the future. You can use the cable supplied with your monitor for this connection if one was provided with its' computer end connector cut off, or run wires to an appropriate jack. On the Amdek monitor, remove the small Joystick type connector from the cable supplied with your monitor and use the wire colors shown in the schematic to connect your monitor. Configure this Amdek monitor to be Apple II compatible via its' internal dip switch per the note provided with your monitor. Now find the "HORZ SIZE" (width) control inside of your Amdek monitor & completely remove its' core. You will find that you will need all the width you can muster from your monitor as Timex overscan CRN get full height via use of the internal vertical size control, but the best in width will still leave about an inch or so of unused screen on both edges. Use the monitors' horz phase and/or horz centering controls to center the display on your monitor. If your particular monitor requires positive sync rather than negative sync, use the pvs & phs outputs on the board instead of the nvs & nhs outputs used for an Apple comp. monitor. You will note that the horz & vert syncs are actually tied together on the board. This is because every RGB monitor I have seen so far is very capeable of separating the two syncs from each other, so thier external separation is not only unnec. but a waste and could possibly even cause instabilities!

You will also note that the intensity input to your monitor, if it has one, is grounded. This is because I have found no way of deriving an intensity signal from the TS2068 computer. I believe, short of a new and different custom chip, that it cannot be readily done. (Bo ahead, prove me wrong!) This only results in the fact that any extra bright areas on the screen show as normal brightness. You will find that this is certainly not such to give up for the new benefits you recieve from your RGB monitor....a BEAUTIFUL crystal clear arcade game quality display with absolutely NO video noise, interference, or crawling dots in the colored areas. You can now see things that you simply couldn't see before with your regular tv or composite monitor!

MODIFY YOUR TS2068 BOARD FOR THE RGB INTERFACE SYNC SEPORATOR AS FOLLOWS:

1) Open your computer by removing the 7 screws from the cases' underside, carefully unplug the keyboard flex cable from the PC board and completely remove the top of the case from the computer. Further disassembly is not nec.
2) Cut board trace on far right of computer board where indicated by an "x" below. Use a sharp knife and be careful to only cut this one trace. Install a wire/wrap wire jumper from base (center) terminal of Q5 to feedthrough indicated. Q5 is the unlabeled transistor just to the right of Q4. Carefully check your work in here for shorts & reinstall keyboard cable & case top.



SYNCWARE NEWS

Volume 1 Number 5

Published by SyncWare Co., PO Box 5177, El Monte, CA. 91734 May-June 1984
Publisher/Editor - Fred Nachbaur COPYRIGHT 1983 SyncWare Co.
Associate Editor - Thomas Bent
JOURNAL FOR ELECTRONIC AND OTHER TECHNICAL APPLICATIONS OF T/S COMPUTERS

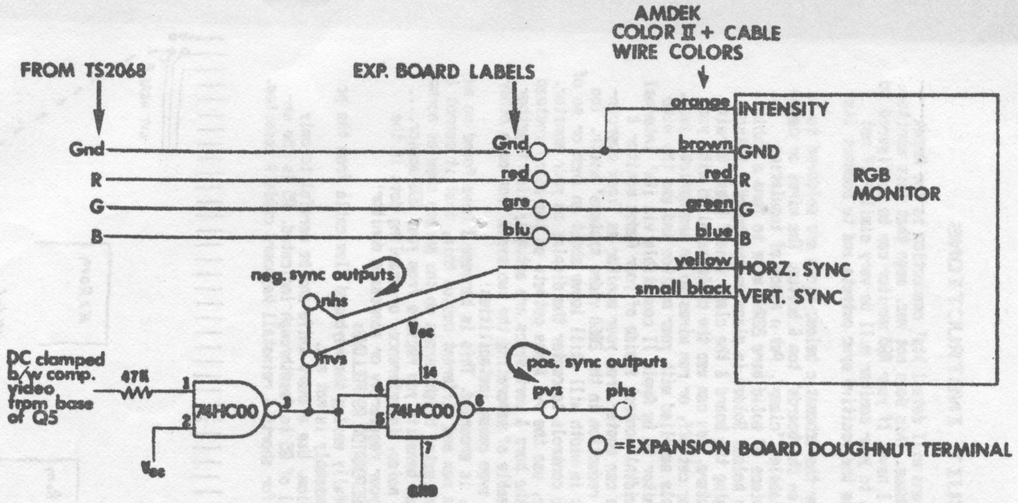
********** DERIVATIVES/VIDEO/VDAQ1**********

TABLE OF CONTENTS

* GOOD NEWS - The "News" Lives! ..... 1 *
* MEET THE "SURVIVORS" ..... 3 *
* MATH DEV. PROGRAM - by Thomas Bent ..... 4 *
* Differentiation I ..... 5 *
* Differentiation II ..... 32 *
* FORTH-WITH Part I BY Gary Smith ..... 8 *
* THE CUSTOM T/S by F.Nachbaur *
* Veni, Vidi Video, Vici ..... 9 *
* A Video Primer ..... 10 *
* VIDREV - an easy video upgrade ..... 14 *
* BUG ALERT! ..... 15 *
* TYD-BYTS ..... 16 *
* PRODUCTS ON PARADE *
* HOT-Z II by Thomas Bent ..... 17 *
* ENER-Z REPORT GENERATOR - P.J. Donnelly ..... 19 *
* DATA ACQUISITION - F.Nachbaur *
* VDAQ1 - A Development Program ..... 21 *
* SWN VOL. 2 QUESTIONNAIRE ..... 35 *
* SWN VOL. 2 ORDER FORM ..... 36 *

SYNCWARE NEWS IS ONLY
$16.95 for a full year of incredible value
Send Your subscription to Thomas Woods
See the article on a previous
Page--DO IT TODAY
AND also subscribe to SYNTAX
And of Course, TSUG Newsletter

USE VISA OR MASTERCHARGE TO ORDER YOUR SWN SUBSCRIPTION



OLIGER 2068 EXPANSION BOARD ASSY INSTRUCTIONS

STEP 1) Using the bare 30AWG wire provided with the board, solder all feedthroughs onto the board by inserting the wire into a feedthrough hole from the top, bending it over onto its' solder pad on the boards' bottom, and then soldering it onto the bottom pad. Then turn the board top up again, cut off the wire about 3/32" from the boards' top surface, bend this end over & solder it in place. Repeat this process until all feedthroughs are soldered in place. A feedthrough is installed at every hole in the board that has a solder pad on the boards' top side (the side marked "made in USA" in the upper left corner), except for the following 9 locations: The 8 outputs for the RGB interface marked with labels such as blu, red, etc. in the upper right hand side of the board, and the pad just below the label R47K on the lower right hand side of the board. This makes 31 feedthrough wires at each end of the board, 5 in the middle part of the board, and two more on the right side of the board--one directly above where the 74HC00 will mount & one directly below this ics' mounting holes. This makes a total of 69 feedthroughs to mount, so plan on spending an hour or longer installing these wires. Clean flux from the board with acetone and a soft cloth and carefully inspect your work for shorts or bad connections. Touch up anything looking suspicious.

STEP 2) If you purchased the parts kit with your expansion board you can skip this step...it's already been done for you! If not, you will need five 35 position double read out edge connectos... 4 of them soldertail and the last soldertail or wirewrap. Prepare these connectors for the board by removing the end pin, both top and bottom, from one end and the 2nd pin from the end, both top and bottom, from the other end of the connector. Now use a fine toothed hacksaw to cut off both ends of the connector right at the slots just vacated by the pins. Use a fine file to smooth each end and you now have a 32 position edge connector. Further prepare this connector by removing 6th pin from the end, both top and bottom, on only one end of your 32 position connector. Now cut a small piece of plastic or pc board materia. to fit the "slot" vacated by the just removed pins & carefully epoxy this into this slot. Repeat this for all five connectors.

STEP 3) Solder the four soldertail edge connectors in place on the expansion board by inserting the connectors pins carefully through the boards' holes while making sure that all the pins actually enter the holes and do not get bent over. Now press the connector down onto the board as far as it will go, bend each pin over on the boards' bottom, and carefully solder in place. Repeat this for all four of the soldertail edge connectors then remove all traces of flux from the board with acetone and a soft cloth or "Q-Tips" and very carfully inspect your work for unsoldered pins and shorts to adjacent traces...the most likely source of problems with this board is at these connections so take extra time and touch up anything looking at all suspicious.

STEP 4) Install the last soldertail or wirewrap edge connector as follows: If it is the wirewrap type then cut all of its' pins down to about 1/4" in length. This connector installs on the end of the board marked "To 2068" with the body of the conn. on the same horz. plane as the board itself and with half its' pins extending over the top edge traces and half over the bottom edge traces of the expansion board. Be sure that the slot on the connector is on the same end as the slots on all the other connectors. Hold the connector so that its' lower pins are laying flat against the edge traces on the non-component side of the board, center the connector up with the edge traces and solder a pin on each end to hold in place. Now solder all the rest of the pins to thier correct edge traces. Now turn over the board and carefully bend each of this sides' pins down to thier edge traces and solder in place. Remove flux with acetone & Q-tips and carefully check your work for shorts.

STEP 5) Install a 47K 1/4 W. resistor at its' marked location on the right of the board, making sure that the end towards the 2068 is soldered on both sides of the board. Install a 14 pin socket on one side of the board and a 20 pin socket on the other side. Carefully insert a 74HC00 ic into the 14 pin socket. (Use appropriate handling with this ic as it is cmos and can be

```
(Program by P. Hoffmann)

250 STOP
260 REM
270 REM
900 REM THE FIVE LOCATIONS THAT
        ARE EXCEPTIONS, ARE
        ACTUALLY OPCODES 125 OR
        126 AND NOT PART OF AN
        ADDRESS OF A CALL OR
        JUMP.
910 REM
1000 IF I=32072 THEN RETURN
1010 IF I=32143 THEN RETURN
1020 IF I=32235 THEN RETURN
1030 IF I=32242 THEN RETURN
1040 IF I=32355 THEN RETURN
1050 IF I=32365 THEN RETURN
1100 LET L=PEEK (I-1)+(256*PEEK
I)+DIFF
1110 POKE J,INT (L/256)
1120 POKE J-1,L-256*INT (L/256)
1130 LET F=1
1140 RETURN
```

SPECTRUM EMULATOR OR CHAMELEON

YES it is here. The ultimate accessory: The Spectrum Emulator. For $60.00 (payable to Doug Dewey) you can have Spectrum Emulation in your home before most other 2068 owners have even mastered the keyboard. Many others are wandering about trying to do this cheaply, importing ROMs from England, breaking into the interior of their 2068 computer, generally messing things up. Still others are trying to do this all in software at a high price--silly fools. There is but one EMU-1 and but one EMU-2. Write for details today. You will be amazed at how level-headed an answer you will get as to the pros and cons of buying this EMULATOR, the availability of various types of software, other options. It is a classic answer that needs great thought and intelligent digestion before appropriate action. The Emulator I offer is THE Emulator--make no bones about it--and will probably run 99.9% of all Spectrum software. But I wouldn't mislead anyone and have them be anything but very satisfied customers, so I claim 95% to 98% of all Spectrum software will work. Remember my motto: Use Common Sense before you buy. We will run a long article in the next newsletter once we have explored all options; in the meantime, Doug Dewey of ENTER LENGUA (and TSUG) fame, can help you. 206 James St., Carrbor, NC 27510--vacation last week of July--919-929-3079.

LARRY G. COUNCILMAN (OWNER) 919-226-0496

AC/DC ELECTRONICS
SEMICONDUCTORS

CAPACITORS
CONNECTORS
RESISTORS
RELAYS

WE BUY
ELECTRONIC COMPONENTS
SURPLUS STOCK
INDIVIDUAL ITEMS

RT. 8, BOX 320
BURLINGTON, N.C. 27215

Large order or small, from good prices on 27128 chips to great deals on disc drives, 1200 baud modems--brand new, with documentation--Larry Councilman of AC/DC Electronics is your one stop shop! Larry has voltage regulators, sockets, caps of all types, the rarely found items, specialty items. He stocks so much that he sells to the big advertisers like JDR, etc. You'll enjoy a visit to his place if you are the least bit into hardware. AC/DC is All the best of a HAMfest--all year long!!

Triangle Sinclair
User's Group
c/o Douglass Dewey
206 James St.
Carrboro, NC 27510

NONPROFIT ORG.
U.S.POSTAGE
PAID
CARRBORO, N.C.
PERMIT NO.83

Roy D. Wistt
178 Cliff St
Norwich Ct 06360

Jan 86

S

QSAVE REVISITED

by PETER D. HOFFMAN
(of DELPHIC ENTERPRISES fame)
5618 Martinique Dr.
Corpus Christi. TX 78411

[This article is a followup to one published by Peter in SYNC. Speaking of Chapter II or "going belly-up" reminds me that TSUSER is no more.]

As a result of reader questions and some new developments, I feel prompted to confuse the issue with this followup. First, in response to all those cards and letters, the following information will supplement some points mentioned in the original review.

To detune the original QSAVE program for 2000 baud,

```
POKE 32201,22
POKE 32285,41
POKE 32294,38
POKE 32203,114
```

This detuning modification has been of only marginal help to borderline tape recording equipment. Most who have tried it (including yours truly) have ended up purchasing new recorders.

To modify the original QSAVE to save the entire 48K of RAM memory,

```
POKE 32082,0
POKE 32249,0
```

Several readers have also inquired about locating the original QSAVE to other addresses, usually the 8K-16K space. The program shown in Figure 1 will make the transfer and correctly modify the absolute addresses of calls and jumps.

Second, there are two new versions of QSAVE that have been issued since the original review. Some common characteristics of these new versions of QSAVE are:

- -break key capability
- -autorun save capability
- -loads up to E_LINE (not the entire 16K as in QSAVE I)
- -programs for 16K and 64K computers

The 16K and 64K programs differ only in where the machine code is located, not in the amount of memory saved. As indicated, the memory is saved up to E_LINE, similar to the SINCLAIR SAVE.

QSAVE II can be identified by the fact that it also has the capability to SAVE/LOAD programs by "name," also similar to the SINCLAIR procedures. QSAVE III does not have the named program facility, only the break key and autorun features.

QSAVE II has a problem in being too critical on timing between the save and load machine code routines. Consequently, it is very unforgiving of less-than-perfect recordings or playback. The reliability can be improved by

```
POKE 8808,128 (64K setup)
or   POKE 32360,128 (16K setup)
```

This does take away the break key capability during the actual program load. You can still break during the lead-in. QSAVE II also does not test the checksum recorded on the tape during "load with name." Consequently, you can't tell if the load was valid.

QSAVE III was apparently released after problems became evident with QSAVE II. It is much more reliable, but has fewer features. The 64K program has one flagrant error that results in an invalid verify every time (even though it was a good save) and causes the program to crash when using the autorun feature. To correct this error do a,

```
POKE 8566,159
POKE 8567,33
```

I certainly hope that there is a fourth version of QSAVE in the works. If there isn't, there needs to be.

Figure 1

```
10 REM PROGRAM FOR TRANSFERING
   QSAVE I TO NON-VOLATILE
   MEMORY.
14 REM
15 REM
20 REM PROGRAMMED BY
          PETER D. HOFFMAN
          5618 MARTINIQUE DR.
          CORPUS CHRISTI, TX.
          78411
25 REM
30 REM PROGRAM DESIGNED TO
40 REM    SHIFT THE ORIGINAL QSAVE
          PROGRAM TO ANY NEW
          ADDRESS. IN PARTICULAR,
          IT ALLOWS QSAVE TO BE
          PUT IN NON-VOLATILE
          MEMORY. TYPICALLY
          ADDRESS SPACE 8K TO 16K.
45 REM
50 REM  NOTES: QSAVE WILL TAKE
          389 BYTES.
54 REM
55 REM
100 PRINT "ENTER THE NEW STARTI
NG ADDRESS
1110 INPUT SA
1115 PRINT "SA
1120 LET DIFF=SA-32000
1130 FOR I=32000 TO 32388
1140 LET J=I+DIFF
1150 LET F=0
1160 LET V=PEEK I
1170 IF (V=125) OR (V=126) THEN
GOSUB 1000
1180 IF F=0 THEN POKE J,V
1190 NEXT I
1200 PRINT "RELOCATED USER C
ALLS ARE:"
2110 PRINT "SAVE - PRINT
USR 32363+DIFF
2220 PRINT "VERIFY - PRINT US
R 32377+DIFF
230 PRINT "LOAD - PRINT US
R 32371+DIFF
240 PRINT "BE SURE AND WRIT
E THESE DOWN."
```

(continued on last page)